C++

Book 1 : Learning
Book 2 : Implementation
Book 3 : Magic & Madness

This documentation is for reference purpose only and is for those who have attended the classroom sessions at Thinking Machines.

- **During your classroom session appropriate theory needs to be written against each example.**
- **You are required to bring this book daily for your classroom sessions.**
- **Some examples won't compile. They have been written to explain some rules.**
- **If you try to understand the examples without attending theory sessions then may god help you.**

Note : Book Three Of C++ is only for full course students.

As discussed in the classroom session, I am assuming that you have a folder named as cppeg and you will be creating the source code files in it. You will have to set the value of the environment variable PATH as we used to do it earlier.

Creating Library and linking it

Create a file named as mylib.h with following declaration of the prototypes of some functions

mylib.h

```
int getStringLength(char *);
void toUpperCase(char *);
void toLowerCase(char *);
void copyString(char *,char *);
void reverseString(char *);
void concatenateString(char *,char *);
```

Now create mylib.c with definition of the functions declared in the mylib.h file

mylib.c

```
int getStringLength(char *p)
{
int x;
for(x=0;*p;x++,p++);
return x;
}
void toUpperCase(char *p)
{
while(*p)
{
if(*p>=97 && *p<=122) *p=*p-32;
p++;
}
}
void toLowerCase(char *p)
{
while(*p)
{
if(*p>=65 && *p<=90) *p=*p+32;
p++;
}
}
void copyString(char *p,char *q)
{
for(;*q;p++,q++) *p=*q;
*p='\0';
}
void concatenateString(char *p,char *q)
{
copyString(p+getStringLength(p),q);
}
void reverseString(char *p)
```

```
{
char *q;
char g;
for(q=p+getStringLength(p)-1;p<q;p++,q--)
{
g=*p;
*p=*q;
*q=g;
}
}
```

To compile the above code and create a library file, follow the following steps
To create object file
gcc -c mylib.c
To create the library file
ar rcs mylib.lib mylib.o

see the contents of the folder using the dir command and you should see the mylib.lib file.

Now let us create a file in which we will be placing call to the functions that are part of our library file mylib.lib

eg1.c (will compile)

```
#include "mylib.h"
#include<stdio.h>
int main()
{
char a[21],b[21];
copyString(a,"Ujjain");
printf("%s\n",a);
copyString(b,"Indore");
printf("%s\n",b);
concatenateString(a,b);
printf("%s\n",a);
toUpperCase(a);
printf("%s\n",a);
toLowerCase(b);
printf("%s\n",b);
reverseString(a);
printf("%s\n",a);
return 0;
}
```

To compile the above code type,

gcc -static eg1.c -lmylib -L. -o eg1.exe

Creating main to accept command line arguments
eg2.c (will compile)

```
#include<stdio.h>
int main(int count,char *s[])
{
int x;
printf("Number of command line arguments : %d\n",count);
printf("Name of the program file as argument number 1 : %s\n",s[0]);
if(count==1)
{
printf("No arguments other than the name of the program\n");
}
else
{
printf("Arguments followed by the name of the program\n");
for(x=1;x<count;x++)
{
printf("Argument number %d : %s\n",x+1,s[x]);
}
}
return 0;
}
```

Compile the above code using
gcc eg2.c -o eg2.exe
Now execute it with various input from CLI as shown below and notice the output
eg2
eg2 God is great
eg2 Ujjain is the "City of Gods"
eg2 100 3000 400

Converting string to int
eg3.c (will compile)

```
#include<stdio.h>
#include<stdlib.h>
int main(int cnt,char *str[])
{
int number,sum;
int x;
if(cnt==1)
{
printf("No numbers passed for processing");
return 0;
}
for(sum=0,x=1;x<cnt;x++)
{
number=atoi(str[x]);
```

```
sum+=number;
}
printf("Total is %d\n",sum);
return 0;
}
```

Compile the above code and executing is as discussed in the classroom session

---

Creating our own convertToInt function
eg4.c (will compile)

```
#include<stdio.h>
#include<string.h>
int convertToInt(char *p)
{
char *q;
int e,n;
for(q=p+strlen(p)-1,e=1,n=0;q>=p;q--,e*=10)
{
if(*q<48 || *q>57) return 0;
n=n+(((*q)-48)*e);
}
return n;
}
int main(int cnt,char *str[])
{
int number,sum;
int x;
if(cnt==1)
{
printf("No numbers passed for processing");
return 0;
}
for(sum=0,x=1;x<cnt;x++)
{
number=convertToInt(str[x]);
sum+=number;
}
printf("Total is %d\n",sum);
return 0;
}
```

Converting int to string
eg5.c (will compile)

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
```

```
char a[21],b[11];
int x;
strcpy(a,"Wattage is : ");
x=60;
itoa(x,b,10);
strcat(a,b);
printf("%s",a);
return 0;
}
```

Creating our own function to convert int to string.
eg6.c (will compile)

```
#include<stdio.h>
#include<string.h>
// note our function is not designed to accept the number system part as itoa  does.
void convertIntToString(int number,char *p)
{
int temp;
int numberOfDigits;
int f;
numberOfDigits=0;
for(temp=number;temp>0;temp=temp/10) numberOfDigits++;
p[numberOfDigits]='\0';
for(temp=number,f=numberOfDigits-1;f>=0;f--,temp=temp/10)
{
p[f]=(temp%10)+48;
}
}
int main()
{
char a[21],b[11];
int x;
strcpy(a,"Wattage is : ");
x=60;
convertIntToString(x,b);
strcat(a,b);
printf("%s",a);
return 0;
}
```

Simplicity introduced in C++ for average programmers who think that declaring variables at top, remembering the format specifiers, applying the address operator and typing backslash are difficult things to remember.

eg7.cpp (will compile)

```
#include<iostream>
int main()
{
std::cout<<"Enter first number : ";
```

```
int x;
std::cin>>x;
std::cout<<"Enter another number : ";
int y;
std::cin>>y;
int z;
z=x+y;
std::cout<<"Total is "<<z<<std::endl;
return 0;
}
```

Note : This is our first C++ example, to compile use the following
g++ eg7.cpp -o eg7.exe

---

eg8.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
cout<<"Enter first number : ";
int x;
cin>>x;
cout<<"Enter another number : ";
int y;
cin>>y;
int z;
z=x+y;
cout<<"Total is "<<z<<endl;
return 0;
}
```

Creating namespace
eg9.cpp (will compile)

```
#include<iostream>
using namespace std;
namespace thinking
{
int getBonus()
{
return 35;
}
}
namespace machines
{
int getBonus()
{
return 20;
}
```

```
}
int main()
{
cout<<thinking::getBonus()<<"% bonus will be given to staff of thinking department"<<endl;
cout<<machines::getBonus()<<"% bonus will be given to staff of machines department"<<endl;
return 0;
}
```

Polymorphism

eg10.cpp (will compile)

```
#include<iostream>
using namespace std;
int add(int e,int f)
{
return e+f;
}
int add(int e,int f,int g)
{
return e+f+g;
}
int add(int e,int f,int g,int h)
{
return e+f+g+h;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10,20 and 30 is "<<add(10,20,30)<<endl;
cout<<"Total of 10,20,30 and 40 is "<<add(10,20,30,40)<<endl;
return 0;
}
```

---

eg11.cpp (will not compile)

```
#include<iostream>
using namespace std;
int add(int e,int f)
{
return e+f;
}
void add(int p,int q)
{
cout<<"Total of "<<p<<" and "<<q<<" is "<<p+q<<endl;
}
int main()
{
int x;
x=add(10,20);
cout<<"Total of 10 and 20 is "<<x<<endl;
```

```
add(100,200);
return 0;
}
```

---

eg12.cpp (will not compile)

```
#include<iostream>
using namespace std;
int add(int p,int q)
{
return p+q;
}
float add(float p,float q)
{
return p+q;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10.2 and 20.3 is "<<add(10.2,20.3)<<endl;
return 0;
}
```

---

eg13.cpp (will compile)

```
#include<iostream>
using namespace std;
int add(int p,int q)
{
return p+q;
}
float add(float p,float q)
{
return p+q;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10.2 and 20.3 is "<<add(10.2f,20.3f)<<endl;
return 0;
}
```

Encapsulation

eg14.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
```

```
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.w=60;
g.setWattage(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.w=100;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

---

eg15.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

---

eg16.cpp (will compile)

```
/*
Henceforth, you have to assume that the classes are designed by one programmer,
The prorammer will put the classes and the helper functions in some library and will
give it to other programmer and the other programmer will be writing the main
function
*/
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(-30);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

---

eg17.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
if(e>=0 && e<=240) w=e;
else w=0;
}
int getWattage()
```

```
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(-30);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

---

eg18.cpp (will not compile)

```
/*
Henceforth, you have to assume that we are validating the data before assigning it to
the encapsulated properties (as done in previous example).
I won't be writing the code to validate as it will waste your lot of time and money.
*/
#include<iostream>
using namespace std;
class Rectangle
{
int length;
int breadth;
public:
void setLength(int e)
{
length=e;
}
void setBreadth(int e)
{
breadth=e;
}
int getLength()
{
return length;
}
int getBreadth()
{
return breadth;
}
};
class Box:public Rectangle
{
int height;
```

```
public:
void setHeight(int e)
{
height=e;
}
int getHeight()
{
return height;
}
};
int main()
{
Box x;
x.length=10;
x.breadth=3;
x.height=40;
x.setLength(10);
x.setBreadth(3);
x.setHeight(40);
cout<<"Length : "<<x.getLength()<<endl;
cout<<"Breadth : "<<x.getBreadth()<<endl;
cout<<"Height : "<<x.getHeight()<<endl;
return 0;
}
```

---

eg19.cpp (will compile)

```
#include<iostream>
using namespace std;
class Rectangle
{
int length;
int breadth;
public:
void setLength(int e)
{
length=e;
}
void setBreadth(int e)
{
breadth=e;
}
int getLength()
{
return length;
}
int getBreadth()
{
```

```
return breadth;
}
};
class Box:public Rectangle
{
int height;
public:
void setHeight(int e)
{
height=e;
}
int getHeight()
{
return height;
}
};
int main()
{
Box x;
x.setLength(10);
x.setBreadth(3);
x.setHeight(40);
cout<<"Length : "<<x.getLength()<<endl;
cout<<"Breadth : "<<x.getBreadth()<<endl;
cout<<"Height : "<<x.getHeight()<<endl;
return 0;
}
```

---

eg20.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Rectangle
{
int length;
int breadth;
public:
void setLength(int e)
{
length=e;
}
void setBreadth(int e)
{
breadth=e;
}
int getLength()
{
return length;
```

```
}
int getBreadth()
{
return breadth;
}
};
class Box:private Rectangle
{
int height;
public:
void setHeight(int e)
{
height=e;
}
int getHeight()
{
return height;
}
};
int main()
{
Box x;
x.length=10;
x.breadth=3;
x.height=40;
x.setLength(10);
x.setBreadth(3);
x.setHeight(40);
cout<<"Length : "<<x.getLength()<<endl;
cout<<"Breadth : "<<x.getBreadth()<<endl;
cout<<"Height : "<<x.getHeight()<<endl;
return 0;
}
```

---

eg21.cpp (will compile)

```
#include<iostream>
using namespace std;
class Rectangle
{
int length;
int breadth;
public:
void setLength(int e)
{
length=e;
}
void setBreadth(int e)
```

```
{
breadth=e;
}
int getLength()
{
return length;
}
int getBreadth()
{
return breadth;
}
};
class Box:private Rectangle
{
int height;
public:
void setHeight(int e)
{
height=e;
}
int getHeight()
{
return height;
}
};
int main()
{
Box x;
x.setHeight(40);
cout<<"Height : "<<x.getHeight()<<endl;
return 0;
}
```

---

eg22.cpp (will compile)

```
#include<iostream>
using namespace std;
class Rectangle
{
int length;
int breadth;
public:
void setLength(int e)
{
length=e;
}
void setBreadth(int e)
{
```

```
breadth=e;
}
int getLength()
{
return length;
}
int getBreadth()
{
return breadth;
}
};
class Box:private Rectangle
{
int height;
public:
void askInformation()
{
int e,f,g;
cout<<"Enter length : ";
cin>>e;
setLength(e);
cout<<"Enter breadth : ";
cin>>f;
setBreadth(f);
cout<<"Enter height : ";
cin>>height;
}
void printInformation()
{
cout<<"Length : "<<getLength()<<endl;
cout<<"Breadth : "<<getBreadth()<<endl;
cout<<"Height : "<<height<<endl;
}
};
int main()
{
Box x;
x.askInformation();
x.printInformation();
return 0;
}
```

Method overriding
eg23.cpp (will compile)

/* Assume 3 programmers are involved in writing the following code and the code is being distribued in library form. */

```
#include<iostream>
using namespace std;
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you"<<endl;
}
};
class JungleBook:public Movie
{
public:
void reelOne()
{
cout<<"Bagheera finds Mowgli"<<endl;
}
void reelTwo()
{
cout<<"Mowgli kills Sher Khan"<<endl;
}
};
int main()
{
JungleBook j;
cout<<"Size of object j is "<<sizeof(j)<<endl;
j.start();
j.reelOne();
j.interval();
j.reelTwo();
j.end();
return 0;
}
```

eg24.cpp (will compile)

```
/* Assume 3 programmers are involved in writing the following code and the code is
being distribued in library form. */
#include<iostream>
using namespace std;
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you"<<endl;
}
};
class JungleBook:public Movie
{
public:
void interval()
{
cout<<"Interval - Have Coke for Rs.25/-"<<endl;
}
void reelOne()
{
cout<<"Bagheera finds Mowgli"<<endl;
}
void reelTwo()
{
cout<<"Mowgli kills Sher Khan"<<endl;
}
};
int main()
{
JungleBook j;
cout<<"Size of object j is "<<sizeof(j)<<endl;
j.start();
j.reelOne();
j.interval();
j.reelTwo();
j.end();
return 0;
```

```
}
```

---

eg25.cpp (will compile)

/* Assume 3 programmers are involved in writing the following code and the code is being distribued in library form.*/

```
#include<iostream>
using namespace std;
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you"<<endl;
}
};
class JungleBook:public Movie
{
public:
void interval()
{
interval();
cout<<"and have Coke for Rs.25/-"<<endl;
}
void reelOne()
{
cout<<"Bagheera finds Mowgli"<<endl;
}
void reelTwo()
{
cout<<"Mowgli kills Sher Khan"<<endl;
}
};
int main()
{
JungleBook j;
cout<<"Size of object j is "<<sizeof(j)<<endl;
j.start();
j.reelOne();
j.interval();
```

```
j.reelTwo();
j.end();
return 0;
}
```

---

eg26.cpp (will compile)

```
/* Assume 3 programmers are involved in writing the following code and the code is
being distribued in library form.*/
#include<iostream>
using namespace std;
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you"<<endl;
}
};
class JungleBook:public Movie
{
public:
void interval()
{
Movie::interval();
cout<<"and have Coke for Rs.25/-"<<endl;
}
void reelOne()
{
cout<<"Bagheera finds Mowgli"<<endl;
}
void reelTwo()
{
cout<<"Mowgli kills Sher Khan"<<endl;
}
};
int main()
{
JungleBook j;
cout<<"Size of object j is "<<sizeof(j)<<endl;
```

```
j.start();
j.reelOne();
j.interval();
j.reelTwo();
j.end();
return 0;
}
```

Access specifier : protected
eg27.cpp (will not compile)

```
class aaa
{
private:
int x;
protected:
int y;
public:
int z;
};
class bbb:public aaa
{
public:
void sam()
{
x=10;
y=20;
z=30;
}
};
class ddd
{
public:
void tom()
{
aaa a;
a.x=10;
a.y=20;
a.z=30;
}
};
int main()
{
aaa a;
a.x=10;
a.y=20;
a.z=30;
return 0;
```

```
}
```

eg28.cpp (will not compile)

```
class aaa
{
private:
int x;
protected:
int y;
public:
int z;
};
class bbb:protected aaa
{

};
class ccc:public bbb
{
public:
void sam()
{
y=20;
z=30;
}
};
class ddd:private aaa
{
};
class eee:public ddd
{
public:
void tom()
{
y=20;
z=30;
}
};
int main()
{
return 0;
};
```

static method

eg29.cpp (will not compile)

```
#include<iostream>
using namespace std;
class aaa
{
```

```
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
static void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
aaa::sam();
aaa::tom();
aaa a;
a.sam();
a.tom();
return 0;
}
```

---

static property
eg30.cpp(will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
static int p;
public:
void setWattage(int e)
{
w=e;
}
void setPrice(int e)
{
p=e;
}
int getWattage()
{
return w;
}
int getPrice()
{
return p;
}
};
```

```
int main()
{
Bulb g,t;
g.setWattage(60);
g.setPrice(10);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Price is "<<g.getPrice()<<endl;
t.setWattage(100);
t.setPrice(15);
cout<<"Wattage is "<<t.getWattage()<<endl;
cout<<"Price is "<<t.getPrice()<<endl;
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Price is "<<g.getPrice()<<endl;
return 0;
}
```

---

eg31.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
static int p;
public:
void setWattage(int e)
{
w=e;
}
void setPrice(int e)
{
p=e;
}
int getWattage()
{
return w;
}
int getPrice()
{
return p;
}
};
int Bulb::p;
int main()
{
Bulb g,t;
g.setWattage(60);
```

```
g.setPrice(10);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Price is "<<g.getPrice()<<endl;
t.setWattage(100);
t.setPrice(15);
cout<<"Wattage is "<<t.getWattage()<<endl;
cout<<"Price is "<<t.getPrice()<<endl;
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Price is "<<g.getPrice()<<endl;
return 0;
}
```

---

call by value
eg32.cpp (will compile)

```
#include<iostream>
using namespace std;
void lmn(int p)
{
p=100;
}
int main()
{
int x;
x=50;
lmn(x);
cout<<x<<endl;
return 0;
}
```

---

call by reference
eg33.cpp (will compile)

```
#include<iostream>
using namespace std;
void lmn(int *p)
{
*p=100;
}
int main()
{
int x;
x=50;
lmn(&x);
cout<<x<<endl;
return 0;
}
```

New technique of creating alias introduced in C++
This example is also an example of call by reference.
eg34.cpp (will compile)

```
#include<iostream>
using namespace std;
void lmn(int &p)
{
p=100;
}
int main()
{
int x;
x=50;
lmn(x);
cout<<x<<endl;
return 0;
}
```

---

eg35.cpp (will not compile)

```
int main()
{
int x;
int &y;
return 0;
}
```

---

eg36.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x;
int &y=x;
y=20;
cout<<x<<endl;
cout<<y<<endl;
x=100;
cout<<x<<endl;
cout<<y<<endl;
return 0;
}
```

Another way of defining member functions
eg37.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
```

```
int w;
public:
void setWattage(int);
int getWattage();
};
void Bulb::setWattage(int e)
{
w=e;
}
int Bulb::getWattage()
{
return w;
}
int main()
{
Bulb g;
g.setWattage(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
return 0;
}
```

---

friend function
eg38.cpp (will not compile)

```
#include<iostream>
using namespace std;
class TV
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of TV : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of TV is : "<<p<<endl;
}
};
class Fridge
{
private:
int p;
public:
void askInformation()
{
```

```
cout<<"Enter price of Fridge : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of Fridge is : "<<p<<endl;
}
};
int main()
{
TV t;
t.askInformation();
Fridge f;
f.askInformation();
cout<<"Total cost of TV and Fridge is : "<<t.p+f.p<<endl;
return 0;
}
```

---

eg39.cpp (will not compile)

```
#include<iostream>
using namespace std;
class TV
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of TV : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of TV is : "<<p<<endl;
}
};
class Fridge
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of Fridge : ";
cin>>p;
}
void printInformation()
```

```
{
cout<<"Price of Fridge is : "<<p<<endl;
}
};
int getTotalCost(TV &a,Fridge &b)
{
return a.p+b.p;
}
int main()
{
TV t;
t.askInformation();
Fridge f;
f.askInformation();
cout<<"Total cost of TV and Fridge is "<<getTotalCost(t,f)<<endl;
return 0;
}
```

---

eg40.cpp (will not compile)

```
#include<iostream>
using namespace std;
class TV
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of TV : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of TV is : "<<p<<endl;
}
friend int getTotalCost(TV &,Fridge &);
};
class Fridge
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of Fridge : ";
cin>>p;
}
```

```
void printInformation()
{
cout<<"Price of Fridge is : "<<p<<endl;
}
friend int getTotalCost(TV &,Fridge &);
};
int getTotalCost(TV &a,Fridge &b)
{
return a.p+b.p;
}
int main()
{
TV t;
t.askInformation();
Fridge f;
f.askInformation();
cout<<"Total cost of TV and Fridge is "<<getTotalCost(t,f)<<endl;
return 0;
}
```

---

eg41.cpp (will compile)

```
#include<iostream>
using namespace std;
class Fridge;
class TV
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of TV : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of TV is : "<<p<<endl;
}
friend int getTotalCost(TV &,Fridge &);
};
class Fridge
{
private:
int p;
public:
void askInformation()
{
```

```
cout<<"Enter price of Fridge : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of Fridge is : "<<p<<endl;
}
friend int getTotalCost(TV &,Fridge &);
};
int getTotalCost(TV &a,Fridge &b)
{
return a.p+b.p;
}
int main()
{
TV t;
t.askInformation();
Fridge f;
f.askInformation();
cout<<"Total cost of TV and Fridge is "<<getTotalCost(t,f)<<endl;
return 0;
}
```

---

friend class
eg41.cpp (will compile)

```
#include<iostream>
using namespace std;
class TV
{
private:
int p;
public:
void askInformation()
{
cout<<"Enter price of TV : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of TV is : "<<p<<endl;
}
friend class Utility;
};
class Fridge
{
private:
int p;
```

```
public:
void askInformation()
{
cout<<"Enter price of Fridge : ";
cin>>p;
}
void printInformation()
{
cout<<"Price of Fridge is : "<<p<<endl;
}
friend class Utility;
};
class Utility
{
public:
int getTotalCost(TV &a,Fridge &b)
{
return a.p+b.p;
}
int getDifferenceInCost(TV &a,Fridge &b)
{
return (a.p-b.p<0)?b.p-a.p:a.p-b.p;
}
int compareCost(TV &a,Fridge &b)
{
return a.p-b.p;
}
};
int main()
{
TV t;
t.askInformation();
Fridge f;
f.askInformation();
Utility u;
cout<<"Total cost of TV and Fridge is "<<u.getTotalCost(t,f)<<endl;
if(u.compareCost(t,f)==0)
{
cout<<"Cost of TV and Fridge is same"<<endl;
} else if(u.compareCost(t,f)<0)
{
cout<<"Cost of Fridge is more than that of TV by "<<u.getDifferenceInCost(t,f)<<endl;
} else if(u.compareCost(t,f)>0)
{
cout<<"Cost of TV is more than that of Fridge by "<<u.getDifferenceInCost(t,f)<<endl;
}
return 0;
```

```
}
```

Multiple Inheritance
eg43.cpp (will compile)

```
#include<iostream>
#include<string.h>
using namespace std;
class Person
{
private:
char name[21];
int age;
public:
void setName(const char *e)
{
strcpy(name,e);
}
void setAge(int e)
{
age=e;
}
int getAge()
{
return age;
}
/*
This method would be inappropriate as it breaks encapsulation
char * getName()
{
return name;
}
*/
void getName(char *e)
{
strcpy(e,name);
}
};
class Employee
{
private:
int employeeId;
public:
void setEmployeeId(int e)
{
employeeId=e;
}
int getEmployeeId()
```

```
{
return employeeId;
}
};
class Doctor:public Person,public Employee
{
private:
char type[51];
public:
void setType(const char *e)
{
strcpy(type,e);
}
void getType(char *e)
{
strcpy(e,type);
}
};
int main()
{
Doctor d;
d.setName("Sameer");
d.setAge(54);
d.setEmployeeId(2031);
d.setType("Cardiologist");
char n[21],t[51];
int a,e;
d.getName(n);
a=d.getAge();
e=d.getEmployeeId();
d.getType(t);
cout<<"Details of the doctor "<<endl;
cout<<"Name : "<<n<<endl;
cout<<"Age : "<<a<<endl;
cout<<"Employee Id. "<<e<<endl;
cout<<"Type : "<<t<<endl;
return 0;
}
```

Multiple inheritance and problems associated with it
eg44.cpp (will not compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
```

```
{
cout<<"Ujjain"<<endl;
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
class ccc:public aaa
{
public:
void joy()
{
cout<<"Dewas"<<endl;
}
};
class ddd:public bbb,public ccc
{
public:
void john()
{
cout<<"Bhopal"<<endl;
}
};
int main()
{
ddd d;
d.sam();
return 0;
}
```

Virtual Inheritance
eg45.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
};
```

```
class bbb:virtual public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
class ccc:virtual public aaa
{
public:
void joy()
{
cout<<"Dewas"<<endl;
}
};
class ddd:public bbb,public ccc
{
public:
void john()
{
cout<<"Bhopal"<<endl;
}
};
int main()
{
ddd d;
d.sam();
return 0;
}
```

Container

eg46.cpp (will compile)

```
#include<iostream>
#include<string.h>
using namespace std;
class Ink
{
private:
char color[21];
float viscosity;
public:
void setColor(const char *e)
{
strcpy(color,e);
}
void getColor(char *e)
```

```
{
strcpy(e,color);
}
void setViscosity(float e)
{
viscosity=e;
}
float getViscosity()
{
return viscosity;
}
};
class Pen
{
private:
int price;
Ink ink;
public:
void setInk(Ink &e)
{
char c[21];
e.getColor(c);
ink.setColor(c);
ink.setViscosity(e.getViscosity());
}
void getInk(Ink &e)
{
char c[21];
ink.getColor(c);
e.setColor(c);
e.setViscosity(ink.getViscosity());
}
void setPrice(int e)
{
price=e;
}
int getPrice()
{
return price;
}
};
int main()
{
Ink i;
i.setColor("Red");
i.setViscosity(0.8f);
Pen pen;
```

```
pen.setInk(i);
pen.setPrice(50);
char c[21];
int p;
float v;
Ink k;
pen.getInk(k);
k.getColor(c);
v=k.getViscosity();
p=pen.getPrice();
cout<<"Price of pen : "<<p<<endl;
cout<<"Color of ink : "<<c<<endl;
cout<<"Viscosity of ink : "<<v<<endl;
return 0;
}
```

Another approach
eg47.cpp (will compile)

```
#include<iostream>
#include<string.h>
using namespace std;
class Ink
{
private:
char color[21];
float viscosity;
public:
void setColor(const char *e)
{
strcpy(color,e);
}
void getColor(char *e)
{
strcpy(e,color);
}
void setViscosity(float e)
{
viscosity=e;
}
float getViscosity()
{
return viscosity;
}
};
class Pen
{
private:
```

```
int price;
Ink ink;
public:
void setColorOfInk(const char *e)
{
ink.setColor(e);
}
void getColorOfInk(char *e)
{
ink.getColor(e);
}
void setViscosityOfInk(float e)
{
ink.setViscosity(e);
}
float getViscosityOfInk()
{
return ink.getViscosity();
}
void setPrice(int e)
{
price=e;
}
int getPrice()
{
return price;
}
};
int main()
{
Pen pen;
pen.setColorOfInk("Red");
pen.setViscosityOfInk(0.8f);
pen.setPrice(50);
char c[21];
int p;
float v;
pen.getColorOfInk(c);
v=pen.getViscosityOfInk();
p=pen.getPrice();
cout<<"Price of pen : "<<p<<endl;
cout<<"Color of ink : "<<c<<endl;
cout<<"Viscosity of ink : "<<v<<endl;
return 0;
}
```

Parameters with default argument
eg48.cpp

```
#include<iostream>
using namespace std;
int add(int p,int q)
{
return p+q;
}
int add(int p,int q,int r)
{
return p+q+r;
}
int add(int p,int q,int r,int s)
{
return p+q+r+s;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10,20 and 30 is "<<add(10,20,30)<<endl;
cout<<"Total of 10,20,30 and 40 is "<<add(10,20,40,50)<<endl;
return 0;
}
```

---

eg49.cpp(will compile)

```
#include<iostream>
using namespace std;
int add(int p,int q,int r=0,int s=0)
{
return p+q+r+s;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10,20 and 30 is "<<add(10,20,30)<<endl;
cout<<"Total of 10,20,30 and 40 is "<<add(10,20,40,50)<<endl;
return 0;
}
```

---

eg50.cpp (will not compile)

```
#include<iostream>
using namespace std;
int add(int p,int q,int r=0,int s)
{
return p+q+r+s;
}
int main()
{
```

```
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10,20 and 30 is "<<add(10,20,30)<<endl;
cout<<"Total of 10,20,30 and 40 is "<<add(10,20,40,50)<<endl;
return 0;
}
```

Constructors

eg51.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g;
cout<<"Wattage is "<<g.getWattage()<<endl;
return 0;
}
```

---

eg52.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w=0;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
```

```
int main()
{
Bulb g;
cout<<"Wattage is "<<g.getWattage()<<endl;
return 0;
}
```

---

eg53.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void initialize()
{
w=0;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g;
cout<<"Wattage is "<<g.getWattage()<<endl;
return 0;
}
```

This page has been intentionally left blank for lot of theory.

eg54.cpp(will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Wattage is "<<t.getWattage()<<endl;
Bulb k;
cout<<"Wattage is "<<k.getWattage()<<endl;
return 0;
}
```

---

eg55.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
Bulb(int e)
{
w=e; // assume that we have validated the value
}
```

```
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Wattage is "<<t.getWattage()<<endl;
Bulb k(100);
cout<<"Wattage is "<<k.getWattage()<<endl;
return 0;
}
```

---

eg56.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb(int e)
{
w=e; // assume that we have validated the value
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Wattage is "<<t.getWattage()<<endl;
Bulb k(100);
cout<<"Wattage is "<<k.getWattage()<<endl;
```

```
return 0;
}
```

eg57.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb(int e=0)
{
w=e;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Wattage is "<<t.getWattage()<<endl;
Bulb k(100);
cout<<"Wattage is "<<k.getWattage()<<endl;
return 0;
}
```

eg58.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
Bulb(int e)
{
```

```
w=e; // assume that we have validated the value
}
Bulb(const Bulb &e)
{
w=e.w;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Wattage is "<<t.getWattage()<<endl;
Bulb k(t);
cout<<"Wattage is "<<k.getWattage()<<endl;
return 0;
}
```

---

eg59.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
Bulb(int e)
{
w=e; // assume that we have validated the value
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
```

```
return w;
}
};
int main()
{
Bulb g,t(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
cout<<"Wattage is "<<t.getWattage()<<endl;
Bulb k(t);
cout<<"Wattage is "<<k.getWattage()<<endl;
return 0;
}
```

---

eg60.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb(const Bulb &e)
{
w=e.w;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g;
g.setWattage(100);
cout<<"Wattage is "<<g.getWattage()<<endl;
Bulb t(g);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

Constructors execution sequence in case of inheritance

eg61.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
};
class bbb:public aaa
{
};
int main()
{
bbb k;
return 0;
}
```

---

eg62.cpp(will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class default constructor"<<endl;
}
};
int main()
{
bbb k;
return 0;
}
```

---

eg63.cpp (will compile)

```
#include<iostream>
```

```
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class default constructor"<<endl;
}
bbb(int e)
{
cout<<"Derived class parameterized constructor"<<endl;
}
};
int main()
{
bbb k;
bbb g(10);
return 0;
}
```

---

eg64.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
aaa(int k)
{
cout<<"Base class parameterized constructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
```

```
{
cout<<"Derived class default constructor"<<endl;
}
bbb(int e)
{
cout<<"Derived class parameterized constructor"<<endl;
}
};
int main()
{
bbb k;
bbb g(10);
return 0;
}
```

---

eg65.cpp (will not compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa(int k)
{
cout<<"Base class parameterized constructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class default constructor"<<endl;
}
bbb(int e)
{
cout<<"Derived class parameterized constructor"<<endl;
}
};
int main()
{
bbb k;
bbb g(10);
return 0;
}
```

---

eg66.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa(int k)
{
cout<<"Base class parameterized constructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb():aaa(100)
{
cout<<"Derived class default constructor"<<endl;
}
bbb(int e):aaa(200)
{
cout<<"Derived class parameterized constructor with one parameter"<<endl;
}
bbb(int e,int f):aaa(e)
{
cout<<"Derived class parameterized constructor with two parameters"<<endl;
}
};
int main()
{
bbb k;
bbb g(10);
bbb m(30,40);
return 0;
}
```

---

eg67.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
};
class bbb:public aaa
```

```
{
public:
bbb()
{
cout<<"Derived class default constructor"<<endl;
}
bbb(const bbb &i)
{
cout<<"Derived class copy constructor"<<endl;
}
};
int main()
{
bbb k;
bbb g(k);
return 0;
}
```

---

eg68.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
aaa(const aaa &r)
{
cout<<"Base class copy constructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class default constructor"<<endl;
}
bbb(const bbb &i)
{
cout<<"Derived class copy constructor"<<endl;
}
};
int main()
{
```

```
bbb k;
bbb g(k);
return 0;
}
```

---

eg69.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
aaa(const aaa &r)
{
cout<<"Base class copy constructor"<<endl;
}
};
class bbb:public aaa
{
public:
};
int main()
{
bbb k;
bbb g(k);
return 0;
}
```

---

eg70.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class default constructor"<<endl;
}
};
class bbb:public aaa
{
public:
};
int main()
{
```

```
bbb k;
bbb g(k);
return 0;
}
```

Pointer to an object
eg71.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
int x;
int *p;
p=&x;
*p=60;
cout<<*p<<endl;
Bulb g;
Bulb *k;
k=&g;
g.setWattage(60);
cout<<"Wattage is : "<<k->getWattage()<<endl;
Bulb m;
Bulb *u;
u=&m;
(*u).setWattage(100);
cout<<"Wattage is "<<(*u).getWattage()<<endl;
return 0;
}
```

---

new operator and nameless memory allocation

eg72.cpp (will compile)

```
int main()
{
int j;
j=60;
new int;
return 0;
}
```

---

eg73.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int j;
j=60;
int *p;
p=new int;
*p=100;
cout<<*p<<endl;
return 0;
}
```

---

eg74.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb *b;
b=new Bulb;
b->setWattage(60);
cout<<"Wattage is : "<<b->getWattage()<<endl;
return 0;
```

```
}
```

Dynamic memory allocation
eg75.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x[10],y,t;
for(y=0;y<=9;y++)
{
cout<<"Enter a number : ";
cin>>x[y];
}
for(y=0,t=0;y<=9;y++) t+=x[y];
cout<<"Total is "<<t<<endl;
return 0;
}
```

---

eg76.cpp (will compile but consider it that it won't compile)

```
#include<iostream>
using namespace std;
int main()
{
int y,t,r;
cout<<"Enter your requirement : ";
cin>>r;
if(r<=0)
{
cout<<"Invalid requirement "<<endl;
return 0;
}
int x[r];
for(y=0;y<r;y++)
{
cout<<"Enter a number : ";
cin>>x[y];
}
for(y=0,t=0;y<r;y++) t+=x[y];
cout<<"Total is "<<t<<endl;
return 0;
}
```

---

eg77.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int y,t,r;
```

```
int *x;
cout<<"Enter your requirement : ";
cin>>r;
if(r<=0)
{
cout<<"Invalid requirement "<<endl;
return 0;
}
x=new int[r];
for(y=0;y<r;y++)
{
cout<<"Enter a number : ";
cin>>x[y];
}
for(y=0,t=0;y<r;y++) t+=x[y];
cout<<"Total is "<<t<<endl;
return 0;
}
```

---

eg78.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x;
x=41;
cout<<x<<endl;
x=041;
cout<<x<<endl;
x=0x41;
cout<<x<<endl;
x=0b100001;
cout<<x<<endl;
return 0;
}
```

---

eg79.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x[5];
int *p;
p=x;
cout<<p<<endl;
p++;
cout<<p<<endl;
p=p+2;
```

```
cout<<p<<endl;
p=p-3;
cout<<(unsigned int)p<<endl;
p++;
cout<<(unsigned int)p<<endl;
p=p+2;
cout<<(unsigned int)p<<endl;
return 0;
}
```

---

eg80.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int y,t,r;
int *x,*p;
cout<<"Enter your requirement : ";
cin>>r;
if(r<=0)
{
cout<<"Invalid requirement "<<endl;
return 0;
}
x=new int[r];
for(p=x,y=0;y<r;y++,p++)
{
cout<<"Enter a number : ";
cin>>*p;
}
for(p=x,y=0,t=0;y<r;y++,p++) t+=*p;
cout<<"Total is "<<t<<endl;
return 0;
}
```

---

eg81.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int y,t,r;
int *x;
cout<<"Enter your requirement : ";
cin>>r;
if(r<=0)
{
cout<<"Invalid requirement "<<endl;
return 0;
```

```
}
x=new int[r];
for(y=0;y<r;y++)
{
cout<<"Enter a number : ";
cin>>*(x+y);
}
for(y=0,t=0;y<r;y++) t+=*(x+y);
cout<<"Total is "<<t<<endl;
return 0;
}
```

---

eg82.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
int r;
cout<<"How many bulb's : ";
cin>>r;
if(r<0)
{
cout<<"Invalid requirement"<<endl;
return 0;
}
Bulb *b;
b=new Bulb[r];
int x,y;
for(y=0;y<r;y++)
{
cout<<"Enter wattage : ";
cin>>x;
b[y].setWattage(x);
```

```
}
for(y=0;y<r;y++)
{
cout<<"Wattage is : "<<(b+y)->getWattage()<<endl;
}
return 0;
}
```

Garbage collection
eg83.cpp (will compile)

```
#include<iostream>
using namespace std;
void lmn()
{
int *x;
x=new int;
*x=60;
cout<<*x<<endl;
}
int main()
{
lmn();
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg84.cpp (will compile)

```
#include<iostream>
using namespace std;
void lmn()
{
int *x;
x=new int;
cout<<(unsigned int)x<<endl;
*x=60;
cout<<*x<<endl;
delete x;
cout<<(unsigned int)x<<endl;
}
int main()
{
lmn();
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg85.cpp(will compile)

```
#include<iostream>
using namespace std;
void lmn()
{
int *x;
x=new int[3];
cout<<(unsigned int)x<<endl;
x[0]=10;
x[1]=20;
x[2]=30;
cout<<x[0]<<endl;
cout<<x[1]<<endl;
cout<<x[2]<<endl;
delete [] x;
cout<<(unsigned int)x<<endl;
}
int main()
{
lmn();
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg86.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb *b;
b=new Bulb;
cout<<(unsigned int)b<<endl;
b->setWattage(60);
```

```
cout<<"Wattage is "<<b->getWattage();
delete b;
cout<<(unsigned int)b<<endl;
b=new Bulb[3];
cout<<(unsigned int)b<<endl;
b[0].setWattage(100);
cout<<"Wattage is "<<b[0].getWattage()<<endl;
b[1].setWattage(0);
cout<<"Wattage is "<<b[1].getWattage()<<endl;
b[2].setWattage(60);
cout<<"Wattage is "<<b[2].getWattage()<<endl;
delete [] b;
cout<<(unsigned int)b<<endl;
return 0;
}
```

Base class pointer is capable of storing address of an object created from its derived class
eg87.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb *b;
b=new Bulb;
b->w=60;
b->setWattage(60);
cout<<"Wattage is : "<<b->getWattage()<<endl;
b->ramu();
return 0;
}
```

---

eg88.cpp (will not compile)

```
#include<iostream>
using namespace std;
```

```
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
};
class bbb
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
aaa *a;
a=new bbb;
bbb *b;
b=new aaa;
return 0;
}
```

---

eg89.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
aaa *a;
```

```
a=new bbb;
return 0;
}
```

---

eg90.cpp (will not compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
bbb *b;
b=new aaa;
return 0;
}
```

---

eg91.cpp (will not compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
```

```
};
int main()
{
aaa *p1;
p1=new aaa;
p1->sam();
p1->tom();

bbb *p2;
p2=new bbb;
p2->sam();
p2->tom();

aaa *p3;
p3=new bbb;
p3->sam();
p3->tom();

bbb *p4;
p4=new aaa;

return 0;
}
```

---

eg92.cpp (will not compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
aaa *p;
p=new bbb;
```

```
p->sam();
p->tom();
return 0;
}
```

---

eg93.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
void tom()
{
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
aaa *p;
p=new bbb;
p->sam();
p->tom();
return 0;
}
```

Virtual function

eg94.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
void sam()
{
cout<<"Ujjain"<<endl;
}
virtual void tom()
```

```
{
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Indore"<<endl;
}
};
int main()
{
aaa *p;
p=new bbb;
p->sam();
p->tom();
return 0;
}
```

---

eg95.cpp (will compile)

```
#include<iostream>
using namespace std;
class MarutiALTO
{
// some 1000 properties
public:
void manual()
{
cout<<"Blah blah blah about Maruti ALTO"<<endl;
}
// many more methods
};
class HondaCity
{
// some 1500 properties
public:
void manual()
{
cout<<"Blah blah blah about Honda City"<<endl;
}
// many more methods
};
int main()
{
int ch;
MarutiALTO m;
```

```
HondaCity h;
cout<<"1. Maruti ALTO"<<endl;
cout<<"2. Honda City"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
if(ch>=1 && ch<=2)
{
if(ch==1)
{
m.manual();
}
if(ch==2)
{
h.manual();
}
}
else
{
cout<<"Invalid choice"<<endl;
}
return 0;
}
```

---

eg96.cpp (will compile)

```
#include<iostream>
using namespace std;
class MarutiALTO
{
// some 1000 properties
public:
void manual()
{
cout<<"Blah blah blah about Maruti ALTO"<<endl;
}
// many more methods
};
class HondaCity
{
// some 1500 properties
public:
void manual()
{
cout<<"Blah blah blah about Honda City"<<endl;
}
// many more methods
};
int main()
```

```
{
int ch;
MarutiALTO *m;
HondaCity *h;
cout<<"1. Maruti ALTO"<<endl;
cout<<"2. Honda City"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
if(ch>=1 && ch<=2)
{
if(ch==1)
{
m=new MarutiALTO;
m->manual();
}
if(ch==2)
{
h=new HondaCity;
h->manual();
}
}
else
{
cout<<"Invalid choice"<<endl;
}
return 0;
}
```

---

eg97.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Car
{
};
class MarutiALTO:public Car
{
// some 1000 properties
public:
void manual()
{
cout<<"Blah blah blah about Maruti ALTO"<<endl;
}
// many more methods
};
class HondaCity:public Car
{
// some 1500 properties
```

```
public:
void manual()
{
cout<<"Blah blah blah about Honda City"<<endl;
}
// many more methods
};
int main()
{
int ch;
Car *c;
cout<<"1. Maruti ALTO"<<endl;
cout<<"2. Honda City"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
if(ch>=1 && ch<=2)
{
if(ch==1)
{
c=new MarutiALTO;
}
if(ch==2)
{
c=new HondaCity;
}
c->manual();
}
else
{
cout<<"Invalid choice"<<endl;
}
return 0;
}
```

---

eg98.cpp (will compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
void manual()
{
}
};
class MarutiALTO:public Car
{
// some 1000 properties
```

```
public:
void manual()
{
cout<<"Blah blah blah about Maruti ALTO"<<endl;
}
// many more methods
};
class HondaCity:public Car
{
// some 1500 properties
public:
void manual()
{
cout<<"Blah blah blah about Honda City"<<endl;
}
// many more methods
};
int main()
{
int ch;
Car *c;
cout<<"1. Maruti ALTO"<<endl;
cout<<"2. Honda City"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
if(ch>=1 && ch<=2)
{
if(ch==1)
{
c=new MarutiALTO;
}
if(ch==2)
{
c=new HondaCity;
}
c->manual();
}
else
{
cout<<"Invalid choice"<<endl;
}
return 0;
}
```

Virtual Polymorphism
eg99.cpp (will compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
virtual void manual()
{
}
};
class MarutiALTO:public Car
{
// some 1000 properties
public:
void manual()
{
cout<<"Blah blah blah about Maruti ALTO"<<endl;
}
// many more methods
};
class HondaCity:public Car
{
// some 1500 properties
public:
void manual()
{
cout<<"Blah blah blah about Honda City"<<endl;
}
// many more methods
};
int main()
{
int ch;
Car *c;
cout<<"1. Maruti ALTO"<<endl;
cout<<"2. Honda City"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
if(ch>=1 && ch<=2)
{
if(ch==1)
{
c=new MarutiALTO;
}
if(ch==2)
{
```

```
c=new HondaCity;
}
c->manual();
}
else
{
cout<<"Invalid choice"<<endl;
}
return 0;
}
```

Pure virtual function and abstract class
eg100.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
virtual void manual()=0;
};
int main()
{
Car c;
return 0;
}
```

---

eg101.cpp (will compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
virtual void manual()=0;
};
int main()
{
Car *c;
return 0;
}
```

---

eg102.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
virtual void manual()=0;
};
int main()
```

```
{
Car *c;
c=new Car;
return 0;
}
```

eg103.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
virtual void manual()=0;
};
class Matiz:public Car
{
};
int main()
{
Matiz m;
return 0;
}
```

eg104.cpp (will compile)

```
#include<iostream>
using namespace std;
class Car
{
public:
virtual void manual()=0;
};
class Matiz:public Car
{
public:
void manual()
{
// some code or no code
}
};
int main()
{
Matiz m;
return 0;
}
```

this pointer
eg105.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

(Strictly not for practical)

eg106.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e,Bulb *this)
{
this->w=e;
}
int getWattage(Bulb *this)
{
return this->w;
}
};
int main()
```

```
{
Bulb g,t;
g.setWattage(60,&g);
cout<<"Wattage is "<<g.getWattage(&g)<<endl;
t.setWattage(100,&t);
cout<<"Wattage is "<<t.getWattage(&t)<<endl;
return 0;
}
```

---

eg107.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int w)
{
w=w;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

---

eg108.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int w)
{
this->w=w;
}
```

```
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(60);
cout<<"Wattage is "<<g.getWattage()<<endl;
t.setWattage(100);
cout<<"Wattage is "<<t.getWattage()<<endl;
return 0;
}
```

Destructor
eg109.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Default Constructor"<<endl;
}
~aaa()
{
cout<<"Destructor"<<endl;
}
};
void lmn()
{
aaa a,b,c;
cout<<"Ujjain"<<endl;
}
int main()
{
aaa p,q;
cout<<"Indore"<<endl;
lmn();
cout<<"Dewas"<<endl;
return 0;
}
```

---

eg110.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Default Constructor"<<endl;
}
~aaa()
{
cout<<"Destructor"<<endl;
}
};
int main()
{
aaa a;
aaa *p;
cout<<"Indore"<<endl;
p=new aaa;
cout<<"Dewas"<<endl;
return 0;
}
```

---

eg111.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Default Constructor"<<endl;
}
~aaa()
{
cout<<"Destructor"<<endl;
}
};
int main()
{
aaa a;
aaa *p;
cout<<"Indore"<<endl;
p=new aaa;
cout<<"Dewas"<<endl;
```

```
delete p;
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg112.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
private:
static int x;
int code;
public:
aaa()
{
code=x;
cout<<"Default Constructor"<<endl;
x++;
}
~aaa()
{
cout<<"Destructor of "<<code<<endl;
}
};
int aaa::x=101;
int main()
{
aaa *p;
cout<<"Indore"<<endl;
p=new aaa[3];
cout<<"Dewas"<<endl;
delete p;
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg113.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
private:
static int x;
int code;
public:
aaa()
{
```

```
code=x;
cout<<"Default Constructor"<<endl;
x++;
}
~aaa()
{
cout<<"Destructor of "<<code<<endl;
}
};
int aaa::x=101;
int main()
{
aaa *p;
cout<<"Indore"<<endl;
p=new aaa[3];
cout<<"Dewas"<<endl;
delete [] p;
cout<<"Ujjain"<<endl;
return 0;
}
```

Destructor execution sequence in case of inheritance
eg114.cpp

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class Default Constructor"<<endl;
}
~aaa()
{
cout<<"Base class Destructor"<<endl;
}
};
class bbb:public aaa
{
};
int main()
{
bbb b;
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg115.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class Default Constructor"<<endl;
}
~aaa()
{
cout<<"Base class Destructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class Default Constructor"<<endl;
}
~bbb()
{
cout<<"Derived class Destructor"<<endl;
}
};
int main()
{
bbb b;
cout<<"Ujjain"<<endl;
return 0;
}
```

---

eg116.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class Default Constructor"<<endl;
}
~aaa()
{
cout<<"Base class Destructor"<<endl;
```

```
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class Default Constructor"<<endl;
}
~bbb()
{
cout<<"Derived class Destructor"<<endl;
}
};
int main()
{
bbb *p;
cout<<"Ujjain"<<endl;
p=new bbb;
cout<<"Indore"<<endl;
delete p;
cout<<"Dewas"<<endl;
return 0;
}
```

---

virtual destructor
eg117.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class Default Constructor"<<endl;
}
~aaa()
{
cout<<"Base class Destructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class Default Constructor"<<endl;
```

```
}
~bbb()
{
cout<<"Derived class Destructor"<<endl;
}
};
int main()
{
aaa *p;
cout<<"Ujjain"<<endl;
p=new bbb;
cout<<"Indore"<<endl;
delete p;
cout<<"Dewas"<<endl;
return 0;
}
```

---

eg118.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
public:
aaa()
{
cout<<"Base class Default Constructor"<<endl;
}
virtual ~aaa()
{
cout<<"Base class Destructor"<<endl;
}
};
class bbb:public aaa
{
public:
bbb()
{
cout<<"Derived class Default Constructor"<<endl;
}
~bbb()
{
cout<<"Derived class Destructor"<<endl;
}
};
int main()
{
aaa *p;
```

```
cout<<"Ujjain"<<endl;
p=new bbb;
cout<<"Indore"<<endl;
delete p;
cout<<"Dewas"<<endl;
return 0;
}
```

---

function template

eg119.cpp (will compile)

```
#include<iostream>
using namespace std;
int add(int x,int y)
{
int z;
z=x+y;
return z;
}
float add(float x,float y)
{
float z;
z=x+y;
return z;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10.2 and 20.3 is "<<add(10.2f,20.3f)<<endl;
return 0;
}
```

---

eg120.cpp (will compile)

```
#include<iostream>
using namespace std;
template<class whatever>
whatever add(whatever x,whatever y)
{
whatever z;
z=x+y;
return z;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10.2 and 20.3 is "<<add(10.2f,20.3f)<<endl;
return 0;
}
```

---

eg121.cpp (will compile)

```
#include<iostream>
using namespace std;
template<class whatever1,class whatever2>
whatever1 add(whatever1 x,whatever2 y)
{
whatever1 z;
z=x+y;
return z;
}
int main()
{
cout<<"Total of 10 and 20 is "<<add(10,20)<<endl;
cout<<"Total of 10.2 and 20.3 is "<<add(10.2f,20.3f)<<endl;
cout<<"Total of 10 and 20.5 is "<<add(10,20.5)<<endl;
cout<<"Total of 30.3 and 30 is "<<add(30.3,30)<<endl;
return 0;
}
```

---

class template

eg122.cpp (will compile)

```
#include<iostream>
using namespace std;
int stack[100];
int lowerBound=0;
int upperBound=99;
int top=100;
int isEmpty()
{
return top==upperBound+1;
}
int isFull()
{
return top==lowerBound;
}
void push(int data)
{
if(isFull()) return;
top--;
stack[top]=data;
}
int pop()
{
int data;
if(isEmpty()) return 0;
data=stack[top];
top++;
```

```
return data;
}
int main()
{
push(10);
push(20);
push(30);
push(40);
while(!isEmpty())
{
cout<<pop()<<endl;
}
return 0;
}
```

---

eg123.cpp (will compile)

```
#include<iostream>
using namespace std;
class Stack
{
private:
int stack[100];
int lowerBound;
int upperBound;
int top;
public:
Stack();
int isEmpty();
int isFull();
void push(int);
int pop();
};
Stack::Stack()
{
lowerBound=0;
upperBound=99;
top=100;
}
int Stack::isEmpty()
{
return top==upperBound+1;
}
int Stack::isFull()
{
return top==lowerBound;
}
void Stack::push(int data)
```

```
{
if(isFull()) return;
top--;
stack[top]=data;
}
int Stack::pop()
{
int data;
if(isEmpty()) return 0;
data=stack[top];
top++;
return data;
}
int main()
{
Stack s1,s2;
s1.push(10);
s1.push(20);
s1.push(30);
s1.push(40);
s2.push(100);
s2.push(200);
s2.push(300);
while(!s1.isEmpty())
{
cout<<s1.pop()<<endl;
}
while(!s2.isEmpty())
{
cout<<s2.pop()<<endl;
}
return 0;
}
```

---

eg124.cpp (will compile)

```
#include<iostream>
using namespace std;
template<class whatever>
class Stack
{
private:
whatever stack[100];
int lowerBound;
int upperBound;
int top;
public:
Stack()
```

```
{
lowerBound=0;
upperBound=99;
top=100;
}
int isEmpty()
{
return top==upperBound+1;
}
int isFull()
{
return top==lowerBound;
}
void push(whatever data)
{
if(isFull()) return;
top--;
stack[top]=data;
}
whatever pop()
{
whatever data;
if(isEmpty()) return 0;
data=stack[top];
top++;
return data;
}
};
int main()
{
Stack<int> s1;
Stack<float> s2;
s1.push(10);
s1.push(20);
s1.push(30);
s1.push(40);
s2.push(100.2f);
s2.push(200.3f);
s2.push(300.4f);
while(!s1.isEmpty())
{
cout<<s1.pop()<<endl;
}
while(!s2.isEmpty())
{
cout<<s2.pop()<<endl;
}
```

```
return 0;
}
```

---

eg125.cpp (will compile)

```
#include<iostream>
using namespace std;
template<class whatever>
class Stack
{
private:
whatever stack[100];
int lowerBound;
int upperBound;
int top;
public:
Stack();
int isEmpty();
int isFull();
void push(whatever);
whatever pop();
};
template<class whatever>
Stack<whatever>::Stack()
{
lowerBound=0;
upperBound=99;
top=100;
}
template<class whatever>
int Stack<whatever>::isEmpty()
{
return top==upperBound+1;
}
template<class whatever>
int Stack<whatever>::isFull()
{
return top==lowerBound;
}
template<class whatever>
void Stack<whatever>::push(whatever data)
{
if(isFull()) return;
top--;
stack[top]=data;
}
template<class whatever>
whatever Stack<whatever>::pop()
```

```
{
whatever data;
if(isEmpty()) return 0;
data=stack[top];
top++;
return data;
}
int main()
{
Stack<int> s1;
Stack<float> s2;
s1.push(10);
s1.push(20);
s1.push(30);
s1.push(40);
s2.push(100.2f);
s2.push(200.3f);
s2.push(300.4f);
while(!s1.isEmpty())
{
cout<<s1.pop()<<endl;
}
while(!s2.isEmpty())
{
cout<<s2.pop()<<endl;
}
return 0;
}
```

Pointer to function
eg126.cpp (will compile)

```
#include<iostream>
using namespace std;
void add(int x,int y)
{
cout<<"Total of "<<x<<" and "<<y<<" is : "<<x+y<<endl;
}
int getSquare(int x)
{
return x*x;
}
int getPower(int base,int exponent)
{
int p,x;
for(p=base,x=2;x<=exponent;x++) p=p*base;
return p;
}
```

```
int main()
{
add(10,20);
cout<<"Square of 5 is "<<getSquare(5)<<endl;
cout<<"5 to the third power or 5 cubed is : "<<getPower(5,3)<<endl;
void (*p)(int,int);
int (*k)(int);
int (*t)(int,int);
p=add;
k=getSquare;
t=getPower;
p(10,20);
cout<<"Square of 5 is "<<k(5)<<endl;
cout<<"5 to the third power or 5 cubed is : "<<t(5,3)<<endl;
return 0;
}
```

Operator overloading
eg127.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
static int x;
int code;
public:
aaa()
{
cout<<"Default value of code "<<code<<endl;
code=x;
cout<<"Default constructor for object with code as "<<code<<endl;
x++;
}
aaa(int e)
{
cout<<"Default value of code "<<code<<endl;
code=x;
cout<<"Parameterized constructor for object with code as "<<code<<endl;
x++;
}
aaa(const aaa &a)
{
cout<<"Default value of code "<<code<<endl;
code=x;
cout<<"code of argument : "<<a.code<<endl;
cout<<"Copy constructor for object with code as "<<code<<endl;
x++;
```

```
}
int getCode()
{
return code;
}
};
int aaa::x=1001;
int main()
{
cout<<"**************** a1 ****************"<<endl;
aaa a1;
cout<<"**************** a2 ****************"<<endl;
aaa a2(10);
cout<<"**************** a3 ****************"<<endl;
aaa a3(a2);
cout<<"**************** a4 ****************"<<endl;
aaa a4=a3;
cout<<"**************** a5 ****************"<<endl;
aaa a5;
a5=10;
cout<<"Code of object named as a5 is "<<a5.getCode()<<endl;
cout<<"**************** m ****************"<<endl;
aaa m;
m=aaa(20);
cout<<"Code of object named as m is "<<m.getCode()<<endl;
cout<<"**************** k ****************"<<endl;
aaa k;
k=m;
cout<<"Code of object named as k is "<<k.getCode()<<endl;
cout<<"**************** j ****************"<<endl;
aaa j;
j=aaa(k);
cout<<"Code of object named as j is "<<j.getCode()<<endl;
return 0;
}
```

---

eg128.cpp (will compile)

```
#include<iostream>
using namespace std;
class aaa
{
static int x;
int code;
public:
aaa()
{
cout<<"Default value of code "<<code<<endl;
```

```
code=x;
cout<<"Default constructor for object with code as "<<code<<endl;
x++;
}
aaa(int e)
{
cout<<"Default value of code "<<code<<endl;
code=x;
cout<<"Parameterized constructor for object with code as "<<code<<endl;
x++;
}
aaa(const aaa &a)
{
cout<<"Default value of code "<<code<<endl;
code=x;
cout<<"code of argument : "<<a.code<<endl;
cout<<"Copy constructor for object with code as "<<code<<endl;
x++;
}
int getCode()
{
return code;
}
void operator=(aaa other)
{
cout<<"= function with parameter details as (aaa other)"<<endl;
cout<<"= function code of argument is "<<other.code<<endl;
cout<<"= function : code of object for which the = is working is "<<code<<endl;
}
void operator=(int i)
{
cout<<"= function with parameter details as (int i)"<<endl;
cout<<"= function : value of argument is "<<i<<endl;
cout<<"= function : code of object for which the = is working is "<<code<<endl;
}
};
int aaa::x=1001;
int main()
{
cout<<"**************** a1 ****************"<<endl;
aaa a1;
cout<<"**************** a2 ****************"<<endl;
aaa a2=a1;
cout<<"**************** a3 ****************"<<endl;
aaa a3;
cout<<"*************** all objects created by code  ****************"<<endl;
a3=a2;
```

```
cout<<"code of object named as a1 is "<<a1.getCode()<<endl;
cout<<"code of object named as a2 is "<<a2.getCode()<<endl;
cout<<"code of object named as a3 is "<<a3.getCode()<<endl;
a3=50;
return 0;
}
```

---

eg129.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
Bulb(const Bulb &other)
{
cout<<"copy constructor got invoked"<<endl;
w=other.w;
}
void operator=(Bulb other)
{
cout<<"= function got invoked"<<endl;
w=other.w;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g;
g.setWattage(60);
cout<<"Wattage of object named as g is "<<g.getWattage()<<endl;
Bulb k(g);
cout<<"Wattage of object named as k is "<<k.getWattage()<<endl;
Bulb m=k;
cout<<"Wattage of object named as m is "<<m.getWattage()<<endl;
```

```
Bulb j;
j=m;
cout<<"Wattage of object named as j is "<<j.getWattage()<<endl;
return 0;
}
```

---

eg130.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x,y,z;
x=y=z=10;
cout<<x<<","<<y<<","<<z<<endl;
return 0;
}
```

---

eg131.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
Bulb(int e)
{
w=e;
}
Bulb(const Bulb &other)
{
cout<<"copy constructor got invoked"<<endl;
w=other.w;
}
void operator=(Bulb other)
{
cout<<"= function got invoked"<<endl;
w=other.w;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
```

```
{
return w;
}
};
int main()
{
Bulb g,t,m;
g=t=m=60;
cout<<"Wattage of bulb named as g is "<<g.getWattage()<<endl;
cout<<"Wattage of bulb named as t is "<<t.getWattage()<<endl;
cout<<"Wattage of bulb named as m is "<<m.getWattage()<<endl;
return 0;
}
```

---

eg132.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
Bulb()
{
w=0;
}
Bulb(int e)
{
w=e;
}
Bulb(const Bulb &other)
{
cout<<"copy constructor got invoked"<<endl;
w=other.w;
}
Bulb & operator=(Bulb other)
{
cout<<"= function got invoked"<<endl;
w=other.w;
return *this;
}
void setWattage(int e)
{
w=e;
}
int getWattage()
{
```

```
return w;
}
};
int main()
{
Bulb g,t,m;
g=t=m=60;
cout<<"Wattage of bulb named as g is "<<g.getWattage()<<endl;
cout<<"Wattage of bulb named as t is "<<t.getWattage()<<endl;
cout<<"Wattage of bulb named as m is "<<m.getWattage()<<endl;
return 0;
}
```

---

eg133.cpp (will compile)

```
#include<stdio.h>
void printNewLineCharacter()
{
printf("\n");
}
class Monitor
{
public:
Monitor & operator<<(int x)
{
printf("%d",x);
return *this;
}
Monitor & operator<<(float x)
{
printf("%f",x);
return *this;
}
Monitor & operator<<(char x)
{
printf("%c",x);
return *this;
}
Monitor & operator<<(const char *x)
{
printf("%s",x);
}
Monitor & operator<<(void (*f)())
{
f();
}
// many many more overloads
};
```

```
namespace tmstd
{
Monitor mout;
void (*newLine)()=printNewLineCharacter;
}
using namespace tmstd;
int main()
{
mout<<"Cool implementation"<<newLine;
mout<<10<<","<<10.2f<<","<<'A'<<newLine;
return 0;
}
```

---

eg134.cpp (will compile)

```
#include<stdio.h>
void printNewLineCharacter()
{
printf("\n");
}
class Monitor
{
public:
Monitor & operator<<(int x)
{
printf("%d",x);
return *this;
}
Monitor & operator<<(float x)
{
printf("%f",x);
return *this;
}
Monitor & operator<<(char x)
{
printf("%c",x);
return *this;
}
Monitor & operator<<(const char *x)
{
printf("%s",x);
}
Monitor & operator<<(void (*f)())
{
f();
}
// many many more overloads
};
```

```
namespace tmstd
{
Monitor mout;
void (*newLine)()=printNewLineCharacter;
}
using namespace tmstd;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
Monitor & operator<<(Monitor &m,Bulb &b)
{
m<<b.getWattage();
return m;
}
int main()
{
Bulb g;
g.setWattage(60);
mout<<"Cool implementation"<<newLine;
mout<<"Wattage is "<<g<<newLine;
return 0;
}
```

---

eg135.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
```

```
{
return w;
}
};
ostream & operator<<(ostream &m,Bulb &b)
{
m<<b.getWattage();
return m;
}
int main()
{
Bulb g;
g.setWattage(60);
cout<<"Predefined Cool implementation"<<endl;
cout<<"Wattage is "<<g<<endl;
return 0;
}
```

---

eg136.cpp (will compile)

```
#include<stdio.h>
class Keyboard
{
public:
void operator>>(int x)
{
scanf("%d",&x);
fflush(stdin);
}
};
namespace tmstd
{
Keyboard kin;
}
using namespace tmstd;
int main()
{
int j;
kin>>j;
printf("%d",j);
return 0;
}
```

---

eg137.cpp (will compile)

```
#include<stdio.h>
class Keyboard
{
public:
void operator>>(int &x)
```

```
{
scanf("%d",&x);
fflush(stdin);
}
void operator>>(float &x)
{
scanf("%f",&x);
fflush(stdin);
}
void operator>>(char &x)
{
scanf("%c",&x);
fflush(stdin);
}
void operator>>(char *x)
{
gets(x);
fflush(stdin);
}
};
namespace tmstd
{
Keyboard kin;
}
using namespace tmstd;
int main()
{
char a[21];
char b;
int c;
float d;
printf("Enter name : ");
kin>>a;
printf("Enter gender : ");
kin>>b;
printf("Enter age : ");
kin>>c;
printf("Enter basic salary : ");
kin>>d;
printf("%s,%c,%d,%f",a,b,c,d);
return 0;
}
```

eg138.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
int main()
{
Bulb g,t;
g.setWattage(60);
t.setWattage(100);
if(g<t)
{
cout<<"Wattage of object named as g is less than that of object named as t"<<endl;
}
else
{
cout<<"Wattage of object named as g is not less than that of object named as t"<<endl;
}
return 0;
}
```

---

eg139.cpp (will compile)

```
#include<iostream>
using namespace std;
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
```

```
return w;
}
int operator<(Bulb &right)
{
return w<right.w;
}
// more functions to handle >  <=  >=  ==  and  !=
};
int main()
{
Bulb g,t;
g.setWattage(60);
t.setWattage(100);
if(g<t)
{
cout<<"Wattage of object named as g is less than that of object named as t"<<endl;
}
else
{
cout<<"Wattage of object named as g is not less than that of object named as t"<<endl;
}
return 0;
}
```

---

eg140.cpp (will compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
Toy(int price)
{
this->price=price;
}
Toy(const Toy &other)
{
this->price=other.price;
}
Toy & operator=(Toy other)
{
this->price=other.price;
```

```
return *this;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
Toy operator+(int number)
{
return Toy(this->price+number);
}
Toy operator+(Toy &other)
{
return Toy(this->price+other.price);
}
// more functions to handle  *  /  % -
};
int main()
{
Toy t1,t2,t3;
t1.setPrice(100);
t2=t1+30;
t3=t1+t2;
cout<<"Price of object named as t1 is "<<t1.getPrice()<<endl;
cout<<"Price of object named as t2 is "<<t2.getPrice()<<endl;
cout<<"Price of object named as t3 is "<<t3.getPrice()<<endl;
return 0;
}
```

---

eg141.cpp (will compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
Toy(int price)
{
this->price=price;
```

```
}
Toy(const Toy &other)
{
this->price=other.price;
}
Toy & operator=(Toy other)
{
this->price=other.price;
return *this;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
void operator+=(int number)
{
this->price=this->price+number;
}
void operator+=(Toy &other)
{
this->price=this->price+other.price;
}
// more functions to handle  -=   *=   /=   %=
};
int main()
{
Toy t1,t2,t3;
t1.setPrice(100);
t2.setPrice(500);
t3.setPrice(1000);
t1+=30;
t3+=t2;
cout<<"Price of object named as t1 is "<<t1.getPrice()<<endl;
cout<<"Price of object named as t2 is "<<t2.getPrice()<<endl;
cout<<"Price of object named as t3 is "<<t3.getPrice()<<endl;
return 0;
}
```

eg142.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
};


class Book
{
private:
int price;
public:
Book()
{
this->price=0;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
};
int operator>(Book &left,Toy &right)
{
return left.price>right.price;
}
int main()
{
```

```
Toy t;
t.setPrice(100);
Book b;
b.setPrice(200);
if(b>t)
{
cout<<"Book b is costlier than Toy t"<<endl;
}
else
{
cout<<"Book b is not costlier than Toy t"<<endl;
}
return 0;
}
```

---

eg143.cpp (will compile)

```
#include<iostream>
using namespace std;
class Book;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
friend int operator>(Book &,Toy &);
};
class Book
{
private:
int price;
public:
Book()
{
this->price=0;
}
```

```
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
friend int operator>(Book &,Toy &);
};
int operator>(Book &left,Toy &right)
{
return left.price>right.price;
}
// more functions to overload relational and arithmetic operators.
int main()
{
Toy t;
t.setPrice(100);
Book b;
b.setPrice(200);
if(b>t)
{
cout<<"Book b is costlier than Toy t"<<endl;
}
else
{
cout<<"Book b is not costlier than Toy t"<<endl;
}
return 0;
}
```

---

eg144.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x,y;
x=10;
x++;
y=20;
++y;
cout<<x<<endl;
cout<<y<<endl;
int z;
z=x++;
cout<<x<<endl;
```

```
cout<<z<<endl;
int p;
p=++y;
cout<<y<<endl;
cout<<p<<endl;
return 0;
}
```

---

eg145.cpp (will not compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
Toy(int price)
{
this->price=price;
}
Toy(const Toy &other)
{
this->price=other.price;
}
Toy & operator=(Toy other)
{
this->price=other.price;
return *this;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
Toy & operator++()
{
this->price=this->price+1;
return *this;
}
};
```

```
int main()
{
Toy t1;
t.setPrice(100);
t1++;
Toy t2;
b.setPrice(200);
++t2;
cout<<"Price of object named as t1 is "<<t1.getPrice()<<endl;
cout<<"Price of object named as t2 is "<<t2.getPrice()<<endl;
return 0;
}
```

---

eg146.cpp (will compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
Toy(int price)
{
this->price=price;
}
Toy(const Toy &other)
{
this->price=other.price;
}
Toy & operator=(Toy other)
{
this->price=other.price;
return *this;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
Toy & operator++()
```

```
{
cout<<"gets invoked on Prefix operation"<<endl;
this->price=this->price+1;
return *this;
}
Toy & operator++(int)
{
cout<<"gets invoked on Postfix operation"<<endl;
this->price=this->price+100;
return *this;
}
//more  functions to overload   – (prefix and postfix)
};
int main()
{
Toy t1;
t1.setPrice(100);
t1++;
Toy t2;
t2.setPrice(100);
++t2;
cout<<"Price of object named as t1 is "<<t1.getPrice()<<endl;
cout<<"Price of object named as t2 is "<<t2.getPrice()<<endl;
return 0;
}
```

---

eg147.cpp (will compile)

```
#include<iostream>
using namespace std;
class IntArrayList
{
private:
int **x;
int bufferSize;
int collectionSize;
int allocatedSize;
public:
IntArrayList()
{
x=new int *[10];
x[0]=new int[10];
bufferSize=10;
collectionSize=0;
allocatedSize=1;
}
IntArrayList(const IntArrayList &other)
{
```

```
x=new int *[other.bufferSize];
for(int r=0;r<other.allocatedSize;r++)
{
x[r]=new int[10];
for(int c=0;c<=9;c++)
{
x[r][c]=other.x[r][c];
}
}
this->bufferSize=other.bufferSize;
this->collectionSize=other.collectionSize;
this->allocatedSize=other.allocatedSize;
}
IntArrayList & operator=(IntArrayList other)
{
for(int e=0;e<allocatedSize;e++)
{
delete [] x[e];
}
delete [] x;
x=new int *[other.bufferSize];
for(int r=0;r<other.allocatedSize;r++)
{
x[r]=new int[10];
for(int c=0;c<=9;c++)
{
x[r][c]=other.x[r][c];
}
}
this->bufferSize=other.bufferSize;
this->collectionSize=other.collectionSize;
this->allocatedSize=other.allocatedSize;
return *this;
}
~IntArrayList()
{
for(int e=0;e<allocatedSize;e++)
{
delete [] x[e];
}
delete [] x;
}
void add(int num)
{
int index=collectionSize%10;
int pointerIndex=collectionSize/10;
if(pointerIndex==allocatedSize)
```

```
{
if(bufferSize==allocatedSize)
{
int **t;
t=new int *[bufferSize+10];
for(int i=0;i<bufferSize;i++)
{
t[i]=x[i];
}
delete [] x;
x=t;
bufferSize+=10;
}
x[pointerIndex]=new int[10];
allocatedSize++;
}
x[pointerIndex][index]=num;
collectionSize++;
}
int get(int i)
{
if(i<0 || i>=collectionSize) return 0;
int index=i%10;
int pointerIndex=i/10;
return x[pointerIndex][index];
}
int operator[](int i)
{
return get(i);
}
int getSize()
{
return collectionSize;
}
};
int main()
{
IntArrayList list1;
cout<<"********** dummy populating starts **********"<<endl;
for(int y=1;y<=34;y++)
{
list1.add(y);
}
cout<<"********** dummy populating ends **********"<<endl;
cout<<"endl<<printing collection in list1"<<endl;
for(int x=0;x<list1.getSize();x++)
{
```

```
cout<<list1[x]<<"  ";
}
cout<<endl<<"Copying to another Array List"<<endl;
IntArrayList list2(list1);
list2.add(35);
cout<<endl<<"printing collection in list2"<<endl;
for(int x=0;x<list2.getSize();x++)
{
cout<<list2[x]<<"  ";
}
cout<<endl<<"printing collection in list1"<<endl;
for(int x=0;x<list1.getSize();x++)
{
cout<<list1[x]<<"  ";
}
IntArrayList list3;
list3.add(5443);
list3.add(1233);
list3.add(1234);
cout<<endl<<"Assigning to another Array List"<<endl;
list3=list2;
list3.add(36);
cout<<endl<<"printing collection in list3"<<endl;
for(int x=0;x<list3.getSize();x++)
{
cout<<list3[x]<<"  ";
}
return 0;
}
```

---

eg148.cpp (will compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
```

```
{
return this->price;
}
friend int operator+(int k,Toy &);
};
int operator+(int k,Toy &e)
{
return k+e.price;
}
int main()
{
Toy t1;
t1.setPrice(100);
int x;
x=200+t1;
cout<<x<<endl;
float f;
f=2.33f+t1;
cout<<f<<endl;
return 0;
}
```

---

eg149.cpp (will compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
operator int()
{
return this->price;
}
};
```

```
int main()
{
Toy t1;
t1.setPrice(100);
int x;
x=200+t1;
cout<<x<<endl;
return 0;
}
```

---

eg150.cpp(will compile)

```
#include<iostream>
using namespace std;
class Toy
{
private:
int price;
public:
Toy()
{
this->price=0;
}
void setPrice(int price)
{
this->price=price;
}
int getPrice()
{
return this->price;
}
operator int()
{
return this->price;
}
};
int main()
{
Toy t1;
t1.setPrice(100);
int x;
x=200+t1;
cout<<x<<endl;
float f;
f=20.3f+t1;
cout<<f<<endl;
return 0;
}
```

File Handling
eg151.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int bold=1;
int italics=2;
int requiredFontSpecification;
cout<<"1. Bold (1)"<<endl;
cout<<"2. Italics (2)"<<endl;
cout<<"3. Bold + Italics (3)"<<endl;
cout<<"Enter your requirement in number : ";
cin>>requiredFontSpecification;
if(requiredFontSpecification==bold)
{
cout<<"Bold required"<<endl;
} else if(requiredFontSpecification==italics)
{
cout<<"Italics required"<<endl;
} else if(requiredFontSpecification==(bold | italics))
{
cout<<"Bold and italics required"<<endl;
} else
{
cout<<"Invalid choice : "<<requiredFontSpecification<<endl;
}
return 0;
}
```

---

eg152.cpp (will compile)

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int x=10;
char a[21];
float y=2.33f;
itoa(x,a,10);
printf("%d as String %s\n",x,a);
sprintf(a,"%d",x);
printf("%d as String %s\n",x,a);
sprintf(a,"%f",y);
printf("%f as string %s\n",y,a);
sprintf(a,"%6.2f",y);
printf("(%6.2f) as string (%s)\n",y,a);
return 0;
```

```
}
```

eg153.cpp (will compile)

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
#define true 1
#define false 0
#define TRUE 1
#define FALSE 0
class TMFileMode
{
private:
TMFileMode()
{
}
public:
static short append;
static short out;
static short in;
static short binary;
};
short TMFileMode::append=1;
short TMFileMode::out=2;
short TMFileMode::in=4;
short TMFileMode::binary=8;
class TMOutputFileStream
{
FILE *f;
int failed;
public:
TMOutputFileStream(const char *fileName)
{
this->failed=false;
this->f=NULL;
this->open(fileName,TMFileMode::out);
}
TMOutputFileStream(const char *fileName,short fileMode)
{
this->failed=false;
this->f=NULL;
this->open(fileName,fileMode);
}
void open(const char *fileName,short fileMode)
{
if(this->f!=NULL)
```

```
{
this->failed=true;
return;
}
if(fileMode==TMFileMode::out)
{
this->f=fopen(fileName,"w");
} else if(fileMode==TMFileMode::append)
{
this->f=fopen(fileName,"a");
} else if(fileMode==(TMFileMode::out | TMFileMode::binary))
{
this->f=fopen(fileName,"wb");
} else if(fileMode==(TMFileMode::append | TMFileMode::binary))
{
this->f=fopen(fileName,"ab");
}
if(f!=NULL) this->failed=false;  else this->failed=true;
}
int hasOperationFailed()
{
return this->failed;
}
void close()
{
if(this->f==NULL) this->failed=true;
else
{
fclose(f);
this->failed=false;
}
}
void operator<<(int x)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
char data[21];
sprintf(data,"%d",x);
fputs(data,this->f);
this->failed=false;
}
void operator<<(char x)
{
if(this->f==NULL)
```

```
{
this->failed=true;
return;
}
fputc(x,this->f);
this->failed=false;
}
void operator<<(const char *x)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
fputs(x,this->f);
this->failed=false;
}
void writeBytes(const char *baseAddress,int size)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
fwrite(baseAddress,size,1,f);
this->failed=false;
}
};
class TMInputFileStream
{
FILE *f;
int failed;
public:
TMInputFileStream(const char *fileName)
{
this->failed=false;
this->f=NULL;
this->open(fileName,TMFileMode::in);
}
TMInputFileStream(const char *fileName,short fileMode)
{
this->failed=false;
this->f=NULL;
this->open(fileName,fileMode);
}
void open(const char *fileName,short fileMode)
{
```

```
if(this->f!=NULL)
{
this->failed=true;
return;
}
if(fileMode==TMFileMode::in)
{
this->f=fopen(fileName,"r");
} else if(fileMode==(TMFileMode::in | TMFileMode::binary))
{
this->f=fopen(fileName,"rb");
}
if(f!=NULL) this->failed=false;  else this->failed=true;
}
int hasOperationFailed()
{
return this->failed;
}
void close()
{
if(this->f==NULL) this->failed=true;
else
{
fclose(f);
this->failed=false;
}
}
void operator>>(int &x)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
if(feof(this->f))
{
x=0;
this->failed=true;
return;
}
char data[21];
char m;
int i;
i=0;
while(1)
{
m=fgetc(this->f);
```

```
if(feof(this->f)) break;
if(m==' ') break;
data[i]=m;
i++;
}
if(i==0)
{
x=0;
this->failed=true;
return;
}
data[i]='\0';
x=atoi(data);
this->failed=false;
}
void operator>>(char &x)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
if(feof(this->f))
{
x=0;
this->failed=true;
return;
}
x=fgetc(this->f);
if(x==EOF)
{
x=0;
this->failed=true;
return;
}
this->failed=false;
}
void operator>>(char *x)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
if(feof(this->f))
{
*x='\0';
```

```
this->failed=true;
return;
}
int i=0;
char m;
while(1)
{
m=fgetc(this->f);
if(feof(this->f)) break;
if(m=='\n') break;
*x=m;
x++;
i++;
}
*x='\0';
if(i==0)
{
this->failed=true;
return;
}
this->failed=false;
}
void readBytes(char *baseAddress,int size)
{
if(this->f==NULL)
{
this->failed=true;
return;
}
if(feof(this->f))
{
for(int x=0;x<size;x++) *(baseAddress+x)=0;
this->failed=true;
return;
}
fread(baseAddress,size,1,f);
if(feof(this->f))
{
this->failed=true;
return;
}
this->failed=false;
}
};
void test1()
{
TMOutputFileStream ofs1("file1.ttt");
```

```
ofs1<<101;
ofs1<<" ";
ofs1<<"Cool Fool";
ofs1<<"\n";
ofs1<<'A';
ofs1<<"Whatever";
ofs1<<"\n";
ofs1.close();
}
void test2()
{
TMInputFileStream ifs1("file1.ttt");
int x;
ifs1>>x;
char n1[101];
ifs1>>n1;
char m;
ifs1>>m;
char n2[101];
ifs1>>n2;
ifs1.close();
cout<<x<<","<<n1<<","<<m<<","<<n2<<endl;
}
class Student
{
private:
int rollNumber;
char name[21];
public:
void askInformation()
{
cout<<"Enter roll number : ";
cin>>rollNumber;
cin.clear();
fflush(stdin);
cout<<"Enter name : ";
cin>>name;
cin.clear();
fflush(stdin);
}
void printInformation()
{
cout<<"Roll number : "<<rollNumber<<endl;
cout<<"Name : "<<name<<endl;
}
};
void test3()
```

```
{
TMInputFileStream ifs2("file2.ttt",TMFileMode::in | TMFileMode::binary);
if(ifs2.hasOperationFailed())
{
cout<<"No students"<<endl;
return;
}
ifs2.close();
}
void test4()
{
char m;
Student student;
while(1)
{
student.askInformation();
TMOutputFileStream ofs2("file2.ttt",TMFileMode::append | TMFileMode::binary);
ofs2.writeBytes((char *)&student,sizeof(Student));
ofs2.close();
cout<<"Student added"<<endl;
cout<<"Add more (Y/N) : ";
cin>>m;
cin.clear();
fflush(stdin);
if(m!='y' && m!='Y') break;
}
}
void test5()
{
TMInputFileStream ifs2("file2.ttt",TMFileMode::in | TMFileMode::binary);
if(ifs2.hasOperationFailed())
{
cout<<"No students"<<endl;
return;
}
Student student;
while(1)
{
ifs2.readBytes((char *)&student,sizeof(Student));
if(ifs2.hasOperationFailed()) break;
student.printInformation();
}
ifs2.close();
}
int main()
{
test1();
```

```
cout<<"Test 1 complete"<<endl;
test2();
cout<<"Test 2 complete"<<endl;
test3();
cout<<"Test 3 complete"<<endl;
test4();
cout<<"Test 4 complete"<<endl;
test5();
cout<<"Test 5 complete"<<endl;
return 0;
}
```

---

eg154.cpp (will compile)

```
#include<stdio.h>
#include<iostream>
#include<fstream>
using namespace std;
void addFriend()
{
char name[21];
int age;
char contactNumber[101];
cout<<"Enter name : ";
cin.getline(name,21); // user might feed space
cin.clear();
fflush(stdin);
cout<<"Enter age : ";
cin>>age;
cin.clear();
fflush(stdin);
cout<<"Enter contact number : ";
cin.getline(contactNumber,21); // user might feed space
cin.clear();
fflush(stdin);
ofstream k;
k.open("friends.frn",ios::app);
k<<name;
k<<"\n";
k<<age;
k<<" ";
k<<contactNumber;
k<<"\n";
k.close();
cout<<"Friend added"<<endl;
}
void displayListOfFriends()
{
```

```
char n[21];
char c[101];
int a;
ifstream g;
g.open("friends.frn");
if(g.fail())
{
cout<<"No contacts"<<endl;
return;
}
while(1)
{
g.getline(n,22);
if(g.fail()) break;
g>>a;
g.getline(c,102);
cout<<"Name : "<<n<<", Age : "<<a<<endl;
cout<<"Contact number : "<<c<<endl;
}
g.close();
}
int main()
{
int ch;
while(1)
{
cout<<"1. Add friend"<<endl;
cout<<"2. Display list of friends"<<endl;
cout<<"3. Exit"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
cin.clear();
fflush(stdin);
if(ch==1) addFriend();
if(ch==2) displayListOfFriends();
if(ch==3) break;
}
return 0;
}
```

---

eg155.cpp (will compile)

```
#include<iostream>
#include<string.h>
#include<stdio.h>
#include<fstream>
using namespace std;
class Employee
```

```
{
private:
int id;
char name[21];
int age;
public:
Employee()
{
id=0;
age=0;
name[0]='\0';
}
Employee(const Employee &other)
{
this->id=id;
strcpy(this->name,other.name);
this->age=age;
}
Employee & operator=(Employee other)
{
this->id=other.id;
strcpy(this->name,other.name);
this->age=other.age;
return *this;
}
void setId(int id)
{
this->id=id;
}
int getId()
{
return this->id;
}
void setName(const char *name)
{
strcpy(this->name,name);
}
void getName(char *name)
{
strcpy(name,this->name);
}
void setAge(int age)
{
this->age=age;
}
int getAge()
{
```

```
return this->age;
}
int operator==(Employee &other)
{
return this->id==other.id;
}
int operator<(Employee &other)
{
return strcmp(this->name,other.name)<0;
}
int operator>(Employee &other)
{
return strcmp(this->name,other.name)>0;
}
int operator<=(Employee &other)
{
return strcmp(this->name,other.name)<=0;
}
int operator>=(Employee &other)
{
return strcmp(this->name,other.name)>=0;
}
int operator!=(Employee &other)
{
return strcmp(this->name,other.name)!=0;
}
};
void addEmployee()
{
int id;
cout<<"Enter employee id. : ";
cin>>id;
cin.clear();
fflush(stdin);
if(id<=0)
{
cout<<"Invalid id. "<<endl;
return;
}
ifstream e1;
e1.open("employees.data");
if(!e1.fail())
{
Employee employee;
while(1)
{
e1.read((char *)&employee,sizeof(employee));
```

```
if(e1.fail()) break;
if(id==employee.getId())
{
e1.close();
char name[21];
employee.getName(name);
cout<<"That id alloted to "<<name<<endl;
return;
}
}
e1.close();
}
char name[21];
int age;
cout<<"Enter name : ";
cin.getline(name,21);
cin.clear();
fflush(stdin);
cout<<"Enter age : ";
cin>>age;
cin.clear();
fflush(stdin);
char m;
cout<<"Save (Y/N) : ";
cin>>m;
cin.clear();
fflush(stdin);
if(m!='y' && m!='Y')
{
cout<<"Employee not added"<<endl;
return;
}
Employee employee;
employee.setId(id);
employee.setName(name);
employee.setAge(age);
ofstream file("employees.data",ios::app | ios::binary);
file.write((char *)&employee,sizeof(Employee));
file.close();
cout<<"Employee added"<<endl;
}
void displayListOfEmployees()
{
ifstream file;
file.open("employees.data",ios::binary);
if(file.fail())
{
```

```
cout<<"No records"<<endl;
return;
}
Employee employee;
int id,age;
char name[21];
while(1)
{
file.read((char *)&employee,sizeof(employee));
if(file.fail()) break;
id=employee.getId();
age=employee.getAge();
employee.getName(name);
cout<<"Id : "<<id<<endl;
cout<<"Name : "<<name<<endl;
cout<<"Age : "<<age<<endl;
}
file.close();
}
int main()
{
int ch;
while(1)
{
cout<<"1. Add employee"<<endl;
cout<<"2. Display list of employees"<<endl;
cout<<"3. Exit"<<endl;
cout<<"Enter your choice : ";
cin>>ch;
cin.clear();
fflush(stdin);
if(ch==1) addEmployee();
if(ch==2) displayListOfEmployees();
if(ch==3) break;
}
return 0;
}
```

Exception Handling
eg156.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x,y,q,r;
cout<<"Enter Dividend : ";
cin>>x;
```

```
cout<<"Enter  Divisor : ";
cin>>y;
q=x/y;
r=x%y;
cout<<"Quotient : "<<q<<endl;
cout<<"Remainder : "<<r<<endl;
if(r==0)
{
cout<<y<<" is factor of "<<x<<endl;
cout<<y<<" x "<<q<<" = "<<x<<endl;
}
else
{
cout<<y<<" is not a factor of "<<x<<endl;
cout<<y<<" x "<<q<<" + "<<r<<" = "<<x<<endl;
}
return 0;
}
```

---

eg157.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x,y,q,r;
cout<<"Enter Dividend : ";
cin>>x;
cout<<"Enter  Divisor : ";
cin>>y;
if(y!=0)
{
q=x/y;
r=x%y;
cout<<"Quotient : "<<q<<endl;
cout<<"Remainder : "<<r<<endl;
if(r==0)
{
cout<<y<<" is factor of "<<x<<endl;
cout<<y<<" x "<<q<<" = "<<x<<endl;
}
else
{
cout<<y<<" is not a factor of "<<x<<endl;
cout<<y<<" x "<<q<<" + "<<r<<" = "<<x<<endl;
}
}
else
```

```
{
cout<<"Divisor cannot be "<<y<<endl;
}
return 0;
}
```

---

eg158.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int x,y,q,r;
cout<<"Enter Dividend : ";
cin>>x;
cout<<"Enter  Divisor : ";
cin>>y;
try
{
if(y==0)
{
throw y;
}
q=x/y;
r=x%y;
cout<<"Quotient : "<<q<<endl;
cout<<"Remainder : "<<r<<endl;
if(r==0)
{
cout<<y<<" is factor of "<<x<<endl;
cout<<y<<" x "<<q<<" = "<<x<<endl;
}
else
{
cout<<y<<" is not a factor of "<<x<<endl;
cout<<y<<" x "<<q<<" + "<<r<<" = "<<x<<endl;
}
}catch(int e)
{
cout<<"Divisor cannot be : "<<e<<endl;
}
return 0;
}
```

---

eg159.cpp (will compile)

```
#include<iostream>
using namespace std;
void divide(int dividend,int divisor,int &quotient,int &remainder)
{
```

```
if(divisor==0)
{
throw divisor;
}
quotient=dividend/divisor;
remainder=dividend%divisor;
}
int main()
{
int x,y,q,r;
cout<<"Enter Dividend : ";
cin>>x;
cout<<"Enter  Divisor : ";
cin>>y;
try
{
divide(x,y,q,r);
cout<<"Quotient : "<<q<<endl;
cout<<"Remainder : "<<r<<endl;
if(r==0)
{
cout<<y<<" is factor of "<<x<<endl;
cout<<y<<" x "<<q<<" = "<<x<<endl;
}
else
{
cout<<y<<" is not a factor of "<<x<<endl;
cout<<y<<" x "<<q<<" + "<<r<<" = "<<x<<endl;
}
}catch(int e)
{
cout<<"Divisor cannot be : "<<e<<endl;
}
return 0;
}
```

List of some predefined exception classes

exception : base class to all exceptions
logic_error inherits exception
the following classes inherit the logic_error
invalid_argument : should be used if the argument is of unacceptable nature
domain_error : should be used if the data or whatever falls outside the module domain
length_error : should be used if the data or whatever exceeds the required length
out_of_range : should be used in case something which is outside the range limits
future_error : is related to multithreading, we will deal with it later on
runtime_error inherits exception
the following classes inherit the runtime_error

range_error : should be used in case the result is cannot be represented in target type
overflow_error : should be used in case the result is going beyond upper limit
underflow_error : should be used in case the result is going lower than the lower limit
regex_error : it is related to regular expressions library, we will deal with it later on
system_error inherits runtime_error
the following classes inherit system_error
ios_base::failure : thrown by functions of input/output library
filesystem::filesystem_error : thrown by functions of file system libray
bad_typeid inherits exception : will generate of typeinfo gets called on a NULL pointer
bad_cast inherits exception : will get generated if casting of reference fails
bad_alloc inherits exception : will be generated if the memory allocataion fails

---

eg160.cpp (will compile)

```
#include<iostream>
using namespace std;
int main()
{
int r;
cout<<"Enter your requirement : ";
cin>>r;
int *x;
try
{
x=new int[r];
int y,t;
for(y=0,t=0;y<r;y++)
{
cout<<"Enter a number : ";
cin>>x[y];
t+=x[y];
}
cout<<"Total is : "<<t<<endl;
delete [] x;
}catch(bad_alloc &e)
{
cout<<"Invalid requirement, unable to allocate memory for "<<r<<" int's"<<endl;
cout<<e.what()<<endl;
}
return 0;
}
```

---

eg161.cpp (will compile)

```
#include<iostream>
#include<typeinfo>
using namespace std;
class aaa
{
public:
```

```
virtual void sam(){}
};
class bbb:public aaa
{
public:
void sam(){}
};
class ccc:public aaa
{
public:
void sam(){}
};
int main()
{
aaa *a=NULL;
int ch;
cout<<"Enter your choice (1. bbb), (2 ccc) : ";
cin>>ch;
if(ch==1)
{
a=new bbb;
}
if(ch==2)
{
a=new ccc;
}
try
{
cout<<typeid(*a).name()<<endl;
}
catch(bad_typeid &e)
{
cout<<e.what()<<endl;
}
return 0;
}
```

---

eg162.cpp (will compile)

```
#include<iostream>
#include<stdexcept>
#include<string.h>
#include<climits>
using namespace std;
class ArithmeticException:public exception
{
private:
char *message;
```

```
public:
ArithmeticException(const char *message)
{
this->message=new char[strlen(message)+1];
strcpy(this->message,message);
}
ArithmeticException(const ArithmeticException &other)
{
this->message=new char[strlen(other.message)+1];
strcpy(this->message,other.message);
}
ArithmeticException & operator=(ArithmeticException other)
{
delete [] message;
this->message=new char[strlen(other.message)+1];
strcpy(this->message,other.message);
return *this;
}
virtual ~ArithmeticException() throw ()
{
delete [] message;
}
virtual const char* what() const throw ()
{
return this->message;
}
};
class Calculator
{
private:
Calculator() {}
public:
static float divide(int x,int y)
{
if(y==0) throw ArithmeticException("/ by zero");
return x/y;
}
static int multiply(int x,int y)
{
int z;
z=x*y;
if(z/y!=x) throw overflow_error("Product will exceed the int limit");
return z;
}
static int add(int x,int y)
{
int z=x+y;
```

```
if(x>0 && y>0 && z<0) throw overflow_error("Sum will exceed int limits");
if(x<0 && y<0 && z>0) throw underflow_error("Sum will go below int limits");
return z;
}
static int substract(int x,int y)
{
int z=x-y;
return z;
}
};
int main()
{
int x,y,z;
try
{
cout<<"Enter first number : ";
cin>>x;
cout<<"Enter second number : ";
cin>>y;
cout<<"Sum : "<<x<<" + "<<y<<" = "<<Calculator::add(x,y)<<endl;
cout<<"Quotient : "<<x<<" / "<<y<<" = "<<Calculator::divide(x,y)<<endl;
cout<<"Product : "<<x<<" x "<<y<<" = "<<Calculator::multiply(x,y)<<endl;
cout<<"Difference : "<<x<<" - "<<y<<" = "<<Calculator::substract(x,y)<<endl;
}catch(ArithmeticException &ae)
{
cout<<ae.what()<<endl;
}
catch(overflow_error &oe)
{
cout<<oe.what()<<endl;
}
return 0;
}
```

---

eg163.cpp (will compile)

```
#include<iostream>
#include<stdexcept>
#include<string.h>
#include<climits>
using namespace std;
class ArithmeticException:public exception
{
private:
char *message;
public:
ArithmeticException(const char *message)
{
```

```
this->message=new char[strlen(message)+1];
strcpy(this->message,message);
}
ArithmeticException(const ArithmeticException &other)
{
this->message=new char[strlen(other.message)+1];
strcpy(this->message,other.message);
}
ArithmeticException & operator=(ArithmeticException other)
{
delete [] message;
this->message=new char[strlen(other.message)+1];
strcpy(this->message,other.message);
return *this;
}
virtual ~ArithmeticException() throw ()
{
delete [] message;
}
virtual const char* what() const throw ()
{
return this->message;
}
};
class Calculator
{
private:
Calculator() {}
public:
static float divide(int x,int y)
{
if(y==0) throw ArithmeticException("/ by zero");
return x/y;
}
static int multiply(int x,int y)
{
int z;
z=x*y;
if(z/y!=x) throw overflow_error("Product will exceed the int limit");
return z;
}
static int add(int x,int y)
{
int z=x+y;
if(x>0 && y>0 && z<0) throw overflow_error("Sum will exceed int limits");
if(x<0 && y<0 && z>0) throw underflow_error("Sum will go below int limits");
return z;
```

```
}
static int substract(int x,int y)
{
int z=x-y;
return z;
}
};
int main()
{
int x,y,z;
try
{
cout<<"Enter first number : ";
cin>>x;
cout<<"Enter second number : ";
cin>>y;
cout<<"Sum : "<<x<<" + "<<y<<" = "<<Calculator::add(x,y)<<endl;
cout<<"Quotient : "<<x<<" / "<<y<<" = "<<Calculator::divide(x,y)<<endl;
cout<<"Product : "<<x<<" x "<<y<<" = "<<Calculator::multiply(x,y)<<endl;
cout<<"Difference : "<<x<<" - "<<y<<" = "<<Calculator::substract(x,y)<<endl;
}catch(ArithmeticException &ae)
{
cout<<ae.what()<<endl;
}
catch(overflow_error &oe)
{
cout<<oe.what()<<endl;
}
catch(...)
{
cout<<"Some problem"<<endl;
}
return 0;
}
```

---

eg164.cpp (will compile)

```
#include<iostream>
#include<ctime>
using namespace std;

int main()
{
char months[12][4]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
char weekDays[7][4]={"Sun","Mon","Tue","Wed","Thu","Fri","Sat"};
time_t t=time(0);
struct tm *now=localtime(&t);
int day=now->tm_mday;
```

```
int month=now->tm_mon+1;
int year=now->tm_year+1900;
int hour=now->tm_hour;
int min=now->tm_min;
int sec=now->tm_sec;
int weekDay=now->tm_wday+1;
int dayOfYear=now->tm_yday+1;
cout<<"Date & time is "<<day<<"/"<<month<<"/"<<year<<", "<<hour<<":"<<min<<":"<<sec<<endl;
cout<<"Month in string form : "<<months[month-1]<<endl;
cout<<"Week day string form : "<<weekDays[weekDay-1]<<endl;
cout<<"Today is "<<dayOfYear<<" day of the year"<<endl;
return 0;
}
```

---

eg165.cpp (will compile)

```
#include<iostream>
#include<ctime>
#include<string.h>
#include<stdio.h>
using namespace std;
class Date
{
private:
int day,month,year;
int hour,minute,second;
int dayOfWeek;
int dayOfYear;
char shortMonth[12][4];
char longMonth[12][10];
char shortWeekDay[7][4];
char longWeekDay[7][10];
Date(int year,int month,int day,int dayOfWeek,int dayOfYear,int hour=0,int minute=0,int second=0)
{
setupStrings();
this->year=year;
this->month=month;
this->day=day;
this->hour=hour;
this->minute=minute;
this->second=second;
this->dayOfWeek=dayOfWeek;
this->dayOfYear=dayOfYear;
}
void setupStrings();
public:
Date()
{
```

```
//setupStrings();
time_t t=time(0);
struct tm *now=localtime(&t);
day=now->tm_mday;
month=now->tm_mon+1;
year=now->tm_year+1900;
hour=now->tm_hour;
minute=now->tm_min;
second=now->tm_sec;
dayOfWeek=now->tm_wday+1;
dayOfYear=now->tm_yday+1;
}
Date(const Date &other)
{
setupStrings();
this->year=other.year;
this->month=other.month;
this->day=other.day;
this->hour=other.hour;
this->minute=other.minute;
this->second=other.second;
this->dayOfWeek=other.dayOfWeek;
this->dayOfYear=other.dayOfYear;
}
Date & operator=(Date other)
{
this->year=other.year;
this->month=other.month;
this->day=other.day;
this->hour=other.hour;
this->minute=other.minute;
this->second=other.second;
this->dayOfWeek=other.dayOfWeek;
this->dayOfYear=other.dayOfYear;
return *this;
}
void getDateString(char *dateString)
{
sprintf(dateString,"%d/%d/%d",this->day,this->month,this->year);
}
void getTimeString(char *timeString)
{
sprintf(timeString,"%d:%d:%d",this->hour,this->minute,this->second);
}
int getDay()
{
return this->day;
```

```
}
int getMonth()
{
return this->month;
}
int getYear()
{
return this->year;
}
int getDayOfWeek()
{
return this->dayOfWeek;
}
int getDayOfYear()
{
return this->dayOfYear;
}
int getHour()
{
return this->hour;
}
int getMinute()
{
return this->minute;
}
int getSecond()
{
return this->second;
}
friend class Calendar;
};
void Date::setupStrings()
{
// setup short month names
strcpy(shortMonth[0],"Jan");
strcpy(shortMonth[1],"Feb");
strcpy(shortMonth[2],"Mar");
strcpy(shortMonth[3],"Apr");
strcpy(shortMonth[4],"May");
strcpy(shortMonth[5],"Jun");
strcpy(shortMonth[6],"Jul");
strcpy(shortMonth[7],"Aug");
strcpy(shortMonth[8],"Sep");
strcpy(shortMonth[9],"Oct");
strcpy(shortMonth[10],"Nov");
strcpy(shortMonth[11],"Dec");
//setup short week day names
```

```
strcpy(shortWeekDay[0],"Sun");
strcpy(shortWeekDay[1],"Mon");
strcpy(shortWeekDay[2],"Tue");
strcpy(shortWeekDay[3],"Wed");
strcpy(shortWeekDay[4],"Thu");
strcpy(shortWeekDay[5],"Fri");
strcpy(shortWeekDay[6],"Sat");
// setup long month names
strcpy(longMonth[0],"January");
strcpy(longMonth[1],"February");
strcpy(longMonth[2],"March");
strcpy(longMonth[3],"April");
strcpy(longMonth[4],"May");
strcpy(longMonth[5],"June");
strcpy(longMonth[6],"July");
strcpy(longMonth[7],"September");
strcpy(longMonth[9],"October");
strcpy(longMonth[10],"November");
strcpy(longMonth[11],"December");
// setup long week day names
strcpy(longWeekDay[0],"Sunday");
strcpy(longWeekDay[1],"Monday");
strcpy(longWeekDay[2],"Tuesday");
strcpy(longWeekDay[3],"Wednesday");
strcpy(longWeekDay[4],"Thursday");
strcpy(longWeekDay[5],"Friday");
strcpy(longWeekDay[6],"Saturday");
}
class Calendar
{
private:
Calendar(){}
static int getDayOfYear(int year,int month,int day)
{
int monthDays[12]={31,28,31,30,31,30,31,31,30,31,30,31};
if((year%4==0 && year%100!=0) || year%400==0) monthDays[1]=29;
int x,y;
for(x=0,y=0;x<month-1;x++) y+=monthDays[x];
y+=day;
return y;
}
static int getDayOfWeek(int year,int month,int day)
{
int baseDay=3; // 1 1 1901 was Tuesday (3)
int baseYear=1901;
int monthDays[12]={31,28,31,30,31,30,31,31,30,31,30,31};
int x;
```

```
int y=0;
for(x=baseYear;x<year;x++)
{
y+=((x%4==0 && x%100!=0) || x%400==0)?366:365;
}
if((year%4==0 && year%100!=0) || year%400==0) monthDays[1]=29;
for(x=0;x<month-1;x++) y+=monthDays[x];
y+=day;
int z=((y%7)+2)%7;  // if start day of week is tue (3,  1-1-1901) then wee ends on Mon(2)
return z;
}
public:
static Date getInstance(int year,int month,int day,int hour=0,int minute=0,int second=0)
{
return
Date(year,month,day,getDayOfWeek(year,month,day),getDayOfYear(year,month,day),hour,minute,sec
ond);
}
};
int main()
{
Date d;
char dateString[21];
char timeString[21];
d.getDateString(dateString);
d.getTimeString(timeString);
cout<<"Now : "<<dateString<<",  "<<timeString<<endl;
Date date2;
date2=Calendar::getInstance(2087,7,29);
date2.getDateString(dateString);
date2.getTimeString(timeString);
cout<<"Now : "<<dateString<<",  "<<timeString<<endl;
cout<<date2.getDayOfWeek();
return 0;
}
```