Practical Programs of C++ for BIT, 2nd Semester
Prepared by: Kumar Poudyal

Basics of C++ programming

// dynamic memory allocation with new and delete

```
#include <iostream>
#include<conio.h>
#include <new>
void main()
{
  int i,n;
  int *p;
  cout << "How many numbers would you like to type? ";
  cin >> n;
 p= new int;
    for (i=0; i<n; i++)
    {
      cout << "Enter number: ";
      cin >> p[i];
    }
    cout << "You have entered: ";
    for (i=0; i<n; i++)
      cout << p[i] << ", ";
    delete p;
    getch();
    }
```

//Passing matrix to function

```
#include<iostream.h>
#include<conio.h>
void disp(int a[][3])
{cout<<endl;
for(int i=0;i<2;i++)
{for(int j=0;j<3;j++)
cout<<a[i][j]<<"\t";
cout<<endl;}}
void main()
{
 int a[2][3]={5,7,1,2,3,4};
 disp(a);
 getch();
 }
```

//Global and Local variable

```
#include<iostream.h>
#include<conio.h>
int a=0;
void main()
{int a=2;
cout<<"Global a:"<<::a<<endl;
::a=5;
cout<<"Local a:"<<a<<endl;
cout<<"Global a:"<<::a<<endl;
getch();
}
```

//Default arguments

```
#include<iostream.h>
#include<conio.h>
float div(int a,int b=2)
{return a/b;}
void main()
{cout<<endl<<"20/5:"<<div(20,5);
cout<<endl<<"Divide 30 by default value:"<<div(30);
getch();
}
```

//function overloading with no. of arguments

```
#include<iostream.h>
#include<conio.h>
int sum(int a,int b,int c)
{return a+b+c;}
int sum(int a,int b)
{return a+b;}
void main()
{int x=7,y=2,z=1;
float e=5,f=3;
cout<<endl<<"Sum of x,y and z:"<<sum(x,y,z);
cout<<endl<<"Sum of e and f:"<<sum(e,f);
cout<<endl<<"Sum of 20,30 and 40:"<<sum(20,30,40);
cout<<endl<<"sum of 2 and 3 is:"<<sum(2,3);
getch();
}
```

//function overloading with different type of arguments

```
#include<iostream.h>
#include<conio.h>
int mul(int a,int b)
{return a*b;}
float mul(float a,float b)
{return a*b;}
void main()
{int c=7,d=2;
float e=5.2,f=2.2;
cout<<endl<<"Product of integers:"<<mul(c,d);
cout<<endl<<"Product of real numbers:"<<mul(e,f);
getch();}
```

//String, Array and Pointer

```
#include<iostream.h>
#include<conio.h>
void disp(char *nam[])
{for(int i=0;i<3;i++)
cout<<endl<<nam[i];
}
void main()
{char *nam[]={"Ravi Gurung","Hari Sharma","Anu Sht"};
disp(nam);
getch();
}
```

//Passing number of strings using array

```
#include<iostream.h>
#include<conio.h>
void disp(char nam[][20])
{for(int i=0;i<3;i++)
cout<<endl<<nam[i];
}
void main()
{char nam[][20]={"Ravi Gurung","Hari Sharma","Anu Sht"};
disp(nam);
getch();
}
```

//Passing string using array

```
#include<iostream.h>
#include<conio.h>
void disp(char nam[])
{cout<<nam;
}
void main()
{char nam[]="Ravi Gurung";
disp(nam);
getch();
}
```

//Passing string using pointer

```
#include<iostream.h>
#include<conio.h>
void disp(char *nam)
{cout<<nam;
}
void main()
{char nam[]="Ravi Gurung";
disp(nam);
getch();
}
```

//Accessing structure variable using pointers

```
#include<iostream.h>
#include<conio.h>
struct student{
     char nam[20];
     int clas;
}s1,*pt;
void main()
{ pt=&s1;
cout<<"Enter Name:";cin.getline(pt->nam,20);
cout<<"Enter Class:";cin>>pt->clas;
cout<<endl<<pt->nam<<endl<<pt->clas;
getch();}
```

//Pointer Arithmetic

```
#include<iostream.h>
#include<conio.h>
void main()
```

```
{
int a[]={1,2,3,4,5};
int *p;
p=a;
for(int i=0;i<5;i++)
cout<<endl<<*(p+i);
getch();
}
```

//return by reference

```
#include<iostream.h>
#include<conio.h>
int& max(int &a,int &b)
{if(a>b)
     return a;
   else return b;}
void main()
{int x=5,y=8;
max(x,y)=0;
cout<<endl<<"x:"<<x<<endl<<"y:"<<y<<endl;
getch();
}
```

//return by pointer

```
#include<iostream.h>
#include<conio.h>
int* max(int *a,int *b)
{if(*a>*b)
     return a;
   else return b;}
void main()
{int x=5,y=8;
cout<<endl<<"The maximum no. is "<<*max(&x,&y);
getch();
}
```

//passing by pointer

```
 #include<iostream.h>
 #include<conio.h>
 void square(int num,int *sq)
 {*sq=num*num;}
 void main(){
 int num=5,sq;
 square(num,&sq);
 cout<<"The square is "<<sq;
 getch();
 }
```

//Passing by Reference

```
#include<iostream.h>
 #include<conio.h>
 void square(int num,int &sq)
 {sq=num*num;}
 void main(){
 int num=5,sq;
```

```
 square(num,sq);
 cout<<"The square is "<<sq;
 getch();
 }
```

//finding sum using recursion

```
#include<iostream.h>
#include<conio.h>
int sum(int num)
{if(num==1)
return 1;
else
return num+sum(num-1);}
void main()
{
int num=10;
cout<<"The sum is "<<sum(num);
getch();
}
```

A program to use get()

```
#include<conio.h>
void main()
{char c;
cin.get(c);
while(c!='\n')
{cout<<c;
cin.get(c);
}
getch();
 }
```

A program to use get() and put()

```
#include<conio.h>
void main()
{char c;
cin.get(c);
while(c!='\n')
{cout.put(c);
cin.get(c);
}
getch();
 }
```

A program to use getline()

```
#include<conio.h>
#include<string.h>
void main()
{
char m[20];
cout<<"What's your name ?";
cin.getline(m,20);
cout<<"Hello "<<m<<"\n";
cout<<"What is your favorite team?";
cin.getline(m,20);
```

```
cout<<"I like "<<m<<" too!\n";
getch();
}
```

//Program using structure

```
#include<iostream.h>
#include<conio.h>
void main()
{
struct student{
int sid;
char nam[20];
char add[20];
}s[10];
for(int i=0;i<2;i++)
{
cout<<"Name:";cin.getline(s[i].nam,20);
 cout<<"Add:";cin.getline(s[i].add,20);
 cout<<"SID:";cin>>s[i].sid;
 cin.ignore(s[i].sid,'\n'); //OR cin.sync();
 }
 for(int i=0;i<2;i++)
{cout<<"Name:"<<s[i].nam<<endl;
 cout<<"Add:"<<s[i].add<<endl;
 cout<<"SID:"<<s[i].sid<<endl;
 } getch();}
```

//Using namespaces

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
namespace digit
{int count=0;};
namespace character
{int count=0;};
namespace other
{int count=0;};
void main()
{
char str[]="1.Nepal 2.India 3.China";
int i=0;
while(str[i]!='\0')
{ if(isdigit(str[i]))
digit::count++;
else if(isalpha(str[i]))
character::count++;
else
other::count++;
i++;}
cout<<"\nNumber of Digits="<<digit::count;
cout<<"\nNumber of Characters="<<character::count;
cout<<"\nNumber of other Characters="<<other::count;
getch();
}
```

//using manipulators

```
#include<iostream.h>
 #include<conio.h>
 #include<string.h>
 #include<iomanip.h>
 void main()
 { int l;
 char nam[5][25]={"Ravi Thapa","Arun Rai","Binita Gurung","Pabitra Sharma","Janak KC"};
 char add[5][25]={"Butwal","Kathmandu","Pokhara","Chitwan","Dang"};
 cout<<"\nName"<<setw(46)<<"Address\n";
 cout<<"\n-----------------------------------------------\n";
 for(int i=0;i<5;i++)
 {l=strlen(nam[i]);
 cout<<endl<<nam[i]<<setw(50-l)<<add[i];}
 getch();
 }
```

//Use of Reference variable

```
#include<iostream.h>
#include<conio.h>
void main()
{
int x=3,&y=x;
cout<<"x="<<x<<endl<<"y="<<y<<endl;
x=x+3;
cout<<"x="<<x<<endl<<"y="<<y<<endl;
y=y+3;
 cout<<"x="<<x<<endl<<"y="<<y<<endl;
 getch();}
```

//Type casting

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a=5,b=2;float z,c=5;
z=a/b;
cout<<a/2<<endl;
cout<<z<<endl;
z=(float)a/b;
cout<<z<<endl;
 z=c/b;
 cout<<z;
 getch();}
```

output

```
2
2
2.5
2.5
```

Classes & Objects

//class and objects

```
//method definition inside the class
#include<iostream.h>
#include<conio.h>
class rectangle
{private:
int length,breadth;
public:
void setdata(int l,int b)
{length =l;
breadth=b;
}
void showdata()
{
cout<<"Length="<<length<<"\nBreadth="<<breadth<<endl; }
int findarea()
{return length*breadth;}
int perimeter()
{ return 2*(length+breadth);}
};
void main()
{
rectangle r;
r.setdata(4,2);
r.showdata();
cout<<"Area= "<<r.findarea()<<endl;
cout<<"Perimeter="<<r.perimeter();getch();
}
```

//class and objects

```
//method definition outside the class
#include<iostream.h>
#include<conio.h>
class rectangle
{private:
int length,breadth;
public:
void setdata(int,int);
void showdata();
int perimeter();
int findarea();
};
void rectangle::setdata(int l,int b)
{length =l;
breadth=b;
}
void rectangle::showdata()
{
cout<<"Length="<<length<<"\nBreadth="<<breadth<<endl; }
int rectangle::findarea()
```

```
{return length*breadth;}
int rectangle::perimeter()
{ return 2*(length+breadth);}
void main()
{
rectangle r;
r.setdata(4,2);
r.showdata();
cout<<"Area= "<<r.findarea()<<endl;
cout<<"Perimeter="<<r.perimeter();getch();
}
```

//Constructors with default argument

```
#include<iostream.h>
#include<conio.h>
class interest
{
int principal,rate,year;
float amount;
public:
interest(int p,int n, int r=10);
void cal();
};
interest::interest(int p, int n, int r)
{principal=p;year=n;rate=r;}
void interest::cal()
{cout<<"Principal="<<principal<<endl;
cout<<"Rate="<<rate<<endl;
cout<<"Year="<<year<<endl;
amount=principal+(float)(principal*year*rate)/100;
cout<<"Amount="<<amount<<endl;}
void main()
{
interest i1(1000,2);
interest i2(1000,2,15);
i1.cal();
i2.cal();
getch();
}
```

//Destructors

```
#include<iostream>
#include<conio.h>
class Marks
{
public:
   int maths;
   int science;
    //constructor
```

```
  Marks() {
    cout<<"Inside Constructor"<<endl;
    cout<<"C++ Object created"<<endl;
  }
   //Destructor
  ~Marks(){
    cout<<"Inside Destructor"<<endl;
    cout<<"C++ Object destructed"<<endl;
  }
};
int main( )
{  {
  Marks m1;
  Marks m2;
   }getch();
   return 0;
  }
```

//dynamic constructors

```
#include<iostream.h>
#include<conio.h>
class list
{
private:
    int *p; // pointer to a list
    int size; // dimension
public:
    list(int x)
    {   size = x;
        p = new int[size];
    }
    void display(int i)
    {
        cout<<p[i]<<"\t";
    }
    void get(int i, int val)
    {
        p[i] = val;
    }
};
void main()
{
    int n;
    cout << "Enter elements in a row\n";
    cin >> n;
    list A(n);
    cout<<"Enter the "<<n<<" numbers:"<<endl;
```

```
    int j, val;
    for (j=0;j<n;j++)
    {
        cin>>val;
        A.get(j,val);
    }
    cout << "\n";
    cout << "List of elements are :\n";
    cout << "----------------------\n";
    for (j=0;j<n;j++)
    A.display(j);
    getch();}
```

//dynamic constructors

```
#include <iostream.h>
#include <conio.h>
class Account
{
private:
int account_no;
int balance;
public :
Account(int a,int b)
{
account_no=a;
balance=b;
}
void display()
{
cout<< "\nAccount number is : "<<account_no;
cout<< "\nBalance is : " <<balance;
}
};

void main()
{
clrscr();
int an,bal;
cout<< "Enter account no : ";
cin>>an;
cout<< "\nEnter balance : ";
cin>>bal;
Account *acc=new Account(an,bal); //dynamic constructor
acc->display(); //'->' operator is used to access the method
getch();
}
```

//Overloading Constructors

```
#include<iostream.h>
#include<conio.h>
class count{
      private:
   int a,b;
   public:
   count()//default constructor
   {a=0;b=0;}
   count(int x,int y)//parameterized constructor
   {a=x;b=y;}
 count(int &x)//copy constructor
  { }
   void display()
   {cout<<endl<<"A="<<a<<endl;
   cout<<"B="<<b<<endl;} };
   void main()
   {count c;
   c.display();
   count d(5,6);
   d.display();
   count &e=d;
   e.display();
   count f=e;//default copy constructor
   f.display();
   getch();            }
```

//Example of const argument

```
#include<iostream.h>
#include<conio.h>
void fun(int &x,const int &y)
{x=100;
//y=200; //can't chang this value
cout<<x<<endl;
cout<<y<<endl;
}
void main()
{
int a=10,b=20;
fun(a,b);
getch();
}
```

//A program to take name and roll no using function and display the data.

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
class student{
private:
char nam[20];int sid;
public:
   int getdata(char n[],int id)
   {strcpy(nam,n);
   sid=id;return 0;}
   int display()
```

```
{
    cout<<"\nName:"<<nam<<endl<<"\nSID="<<sid;return 0;
}
};
int main()
{
    student s1;
    s1.getdata("Ravi",45); clrscr();
    s1.display();
    getch();
    return 0;
}
```

//A program to use parameterized constructor for name and roll no.

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
//using namespace std;
class student{
private:
char nam[20];int sid;
public:
  student(char n[],int id)
  {strcpy(nam,n);
  sid=id;}
  int display()
  {
      cout<<"\nName:"<<nam<<endl<<"\nSID="<<sid;return 0;
  }
  };
  int main()
  {
      student s1("Ravi",45);
      clrscr();
      s1.display();
      getch();
      return 0;
  }
```

//A program to use parameterized constructor for name and roll no.

```
#include<iostream.h>
#include<conio.h>
class student
{ char *nam;int age;
public:
student(char *n,int a)
{nam=n;
age=a;}
void display()
{cout<<"Name:"<<nam<<endl;
cout<<"Age:"<<age<<endl;}};
void main()
{student fs("Steven",21);
```

```
fs.display();
getch();
}
```

Operator Overloading

//overloading preincrementor ++ operator

```
#include <iostream.h>
#include<conio.h>
class Test
{
  private:
    int a,b;
  public:
     Test():a(5),b(6){}
     void operator ++()
     {  a = a+1;
     b = b+1;
     }
     void Display() { cout<<"a: "<<a<<endl<<"b:"<<b<<endl; }
};
int main()
{   Test t;
   ++t;
   t.Display();getch();
   return 0;
}
```

//Overloading binary operator +

```
#include<iostream.h>
#include<conio.h>
class Rectangle
{   int L,B;
    public:
    Rectangle()          //Default Constructor
      {       L = 0;
              B = 0;          }
    Rectangle(int x,int y)          //Parameterize Constructor
      {       L = x;
              B = y;          }
    Rectangle operator+(Rectangle Rec)      //Binary operator overloading func.
      {       Rectangle R;
              R.L = L + Rec.L;
              R.B = B + Rec.B;
              return R;       }
      void Display()
```

```
        {       cout<<"\n\tLength : "<<L;
                cout<<"\n\tBreadth : "<<B;}};
     void main()
     {  Rectangle R1(2,5),R2(3,4),R3;
        //Creating Objects
        cout<<"\n\tRectangle 1 : ";
        R1.Display();
        cout<<"\n\n\tRectangle 2 : ";
        R2.Display();
        R3 = R1 + R2;
        cout<<"\n\n\tRectangle 3 : ";
        R3.Display(); getch();        }
```

//Example of overloading + operator

```
#include <iostream>
using namespace std;
class expense{int rent,fee;
public:
   expense()
   {rent=0;
   fee=0;
   }
   expense(int r,int f){
   rent=r;
   fee=f;}
   expense operator +(expense e)
   {
      expense e2;
      e2.rent=rent+e.rent;
      e2.fee=fee+e.fee;
      return e2;
   }
display(){
cout<<"Rent="<<rent<<endl<<"Fee="<<fee<<endl;}};
int main()
{
   expense Jan(500,100),Feb(500,100),Mar(500,100),total;
   Jan.display();
   Feb.display();
   Mar.display();
   total=Jan+Feb+Mar;
   total.display();
   }
```

// Overloading preincrementer and postincrementer

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

```
class test{private:
   int a,b,c;
public:
   test(){    a=5;    b=5;    c=0;}
   int operator++()
   {      ++a;
      c=++b;}
      int operator++(int)
   {       a++;
      c=b++;}
      int display()   {           cout<<"a="<<a<<endl<<"b="<<b<<endl<<"c="<<c<<endl;return 0;
      }
   };
   int main()
   {
      test t;
      ++t;
      t.display();
      t++;
      t.display();
      return 0;
   }
```

//Overloading comparision operator: <

```
#include<iostream.h>
#include<conio.h>
class time
{int hr;
int min;
public:
void getdata()
{cout<<"Enter hour and minute"<<endl;
cin>>hr>>min;
}
int operator <(time t)
{ int ft,st;//first time and second time
ft=hr*60+min;//convert into minute
st=t.hr*60+t.min;
if(ft<st)
return 1;
else
return 0;
}
};
int main()
{
time t1, t2;
```

```
t1.getdata();
t2.getdata();
if(t1<t2)
cout<<"t1 is less than t2"<<endl;
else
cout<<"t1 is greater or equal to t2"<<endl;
getch();
return 0;}
```

//Overloading Assignment Operator

```
#include<iostream.h>
#include<conio.h>
class marks
{ private:
int m1,m2;
public:
marks(){m1=0;m2=0;}
marks(int i,int j){m1=i;m2=j;}
//overloading assignment operator
void operator =(const marks &m)
{m1=m.m1;
m2=m.m2;
}
void display()
{cout<<"\nMarks in 1st subject:"<<m1;
cout<<"\nMarks in 2nd subject:"<<m2;
}};
void main()
{marks ram(45,89);
marks hari(36,59);
cout<<"\nMarks of Ram:";
ram.display();
cout<<"\nMarks of Hari:";
hari.display();
//use assignment operator
ram=hari;
cout<<"\nMarks of Ram:";
ram.display();
getch();
}
```

//Overloading + operator to join two strings

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class String
```

```
{char *s;
int l;//length of string
public:
void getdata()
{char str[20];
cout<<"Enter a string"<<endl;
cin>>str;
l=strlen(str); s=new char[l+1];
strcpy(s,str);}
void display()
{cout<<s<<endl;}
String operator +(String x)
{
String temp;
temp.s=new char[l+x.l+1];
strcpy(temp.s,s);strcat(temp.s,x.s);
return temp;}};
void main()
{String s1,s2,s3;
s1.getdata();
s2.getdata();
s3=s1+s2;
cout<<"s3=";
s3.display();
getch();}
```

//Overloading + operator to join two strings
//Simple example

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class String
{char s[30];
public:
void getdata()
{cout<<"Enter a string"<<endl;
cin>>s;}
void display()
{cout<<s<<endl;}
String operator +(String x)
{String temp;
strcpy(temp.s,s);strcat(temp.s," ");strcat(temp.s,x.s);
return temp;}};
void main()
{String s1,s2,s3;
s1.getdata();
```

```
 s2.getdata();
 s3=s1+s2;
 cout<<"s3=";
 s3.display();
 getch();}
```

//Example of Type conversion from Basic data type to user defined data type

```
 #include<iostream.h>
 #include<conio.h>
 class distance
 {int feet,inch;
 public:
 distance(float m)
 {feet=int(m);
 inch=12*(m-feet);
 }
 void display()
 {cout<<"Feet="<<feet<<endl<<"Inch="<<inch;}
 };
 void main()
 {
 distance d(2.5);
 d.display();
 getch();
 }
```

```
Output:
Feet=2
Inch=6
```

//Example of Type conversion from user defined data type to basic type

```
 #include<iostream.h>
 #include<conio.h>
 class distance
 {int feet,inch;
 public:
 distance(int f,int i)
 {feet=f;inch=i;}
 float convert()
 {float a=feet+inch/12.0;
 return a;}};
 void main()
 {distance d(5,6);
 cout<<"\nThe distance is "<<d.convert();
 getch();}
```

//Example of Type conversion from user defined data type to basic type using operator function

```
#include<iostream.h>
#include<conio.h>
class distance
{int feet,inch;
public:
distance(int f,int i)
{feet=f;
inch=i;}
operator float()
{float a=feet+inch/12.0;
return a;}};
void main()
{distance d(5,6);float x=(float)d;
cout<<"\nThe distance is "<<d;
getch();}
```

Output:
The distance is 5.5

//Type conversion from object to object or userdefined data type to userdefined type

```
#include<iostream.h>
#include<conio.h>
class distance
{int feet,inch;
public:
distance()
{feet=0;
inch=0;}
distance(int f,int i)
{feet=f;
inch=i;}
void display()
{cout<<endl<<feet<<"ft "<<inch<<"inch"<<endl;}};
class dist{int meter, centi;
public:
dist(int m,int c)
{meter=m;centi=c;}
operator distance()//operator function
{distance d;int f,i;
f=meter*3.3;
i=centi*0.4;
f=f+i/12;
i=i%12;
return distance(f,i);}};
int main()
```

```
{dist d2(4,50);
distance d1;
d1=d2;
d1.display();
getch();}
```

Inheritance

Inheritance

//Example of single inheritance

```
#include<iostream.h>
#include<conio.h>
class student
{ char *nam;int age;
public:
student(char *n,int a)
{nam=n;
age=a;}
void display()
{cout<<"Name:"<<nam<<endl;
cout<<"Age:"<<age<<endl;}};
class fstudent:public student
{char *country;
public:
class fstudent(char *n,int a, char *c):student(n,a)
{country=c;}
void displayf()
{display();
cout<<"Country:"<<country<<endl;}
};
void main()
{fstudent fs("Steven",21,"UK");
fs.displayf();
getch();
}
```

//Example of multiple Inheritance

```
#include<iostream.h>
#include<conio.h>
class teacher{int tid;char subject[20];
public:
void getteacher()
{cout<<"Enter Teacher id and subject"<<endl;
cin>>tid>>subject;
}
```

```
void displayt()
{cout<<"Teacher ID:"<<tid<<endl;
cout<<"Subject:"<<subject<<endl;
}
};
class staff{
int sid;char level[10];
public:
void getstaff()
{cout<<"Enter staff ID and level"<<endl;
cin>>sid>>level;}
void displays()
{ cout<<"StaffID:"<<sid<<endl;
cout<<"Level:"<<level<<endl;
}};
class coordinator:public teacher,public staff
{
char program[20];
public:
void getdata()
{getteacher();
getstaff();
cout<<"Enter Program"<<endl;
cin>>program;
}
void displaydata()
{displayt();
displays();
cout<<"Program:"<<program;
}};
void main()
{coordinator c;
c.getdata();
cout<<"\n----------Coordinator details----------"<<endl;
cout<<"-----------------------------------------"<<endl;
c.displaydata();
getch();}
```

//Example of Hierarchical Inheritance

```
#include<iostream.h>
#include<conio.h>
class employee
{int eid,salary;
public:
void getemp()
{cout<<"Enter id and salary of employee"<<endl;
```

```
cin>>eid>>salary;}
void displayemp()
{cout<<"Emp ID:"<<eid<<endl;
cout<<"Salary:"<<salary<<endl;}};
class engineer:public employee
{char dept[10];
public:
void getdata(){getemp();
cout<<"Enter Department"<<endl;
cin>>dept;}
void display()
{displayemp();
cout<<"Depart:"<<dept<<endl;}};
class typist:public employee
{ int ts;//typing speed
public:
void getdata()
{getemp();
cout<<"Enter typing speed"<<endl;
cin>>ts;}
void display()
{displayemp();
cout<<"Typing speed:"<<ts<<endl;
}};
void main()
{engineer e; typist t;
e.getdata();t.getdata();
cout<<"\nEmployee Details\n";
e.display();cout<<endl;
t.display();
getch();}
```

//Example of Multilevel Inheritance

```
#include<iostream.h>
#include<conio.h>
class student{int roll; char name[20];
public:
void getstudent()
{ cout<<"Enter roll no and name of student"<<endl;
cin>>roll>>name;}
void displaystudent()
{cout<<"Roll No.:"<<roll<<endl;
cout<<"Name:"<<name<<endl;}};
class marks:public student
{ int sub1,sub2,sub3;
public:
```

```
void getmarks()
{cout<<"Enter marks in three subjects"<<endl;
cin>>sub1>>sub2>>sub3;}
void displaymarks()
{cout<<"Subject1:"<<sub1<<endl;
cout<<"Subject2:"<<sub2<<endl;
cout<<"Subject3:"<<sub3<<endl;}
int findtotal()
{return sub1+sub2+sub3;}};
class result:public marks
{float total,percent;
public:
void getdata()
{getstudent();
getmarks();}
void displaydata()
{displaystudent();
displaymarks();
total=findtotal();
percent=total/3;
cout<<"\nTotal Marks:"<<total<<endl;
cout<<"Percentage:"<<percent;}};
void main()
{result r;
r.getdata();
cout<<"\n---Result details----\n";
r.displaydata();getch();}
```

//Derived Class Constructor

```
#include<iostream.h>
#include<conio.h>
class A
{protected:
int adata;
public:
A(int a)
{adata=a;}};
class B:public A
{int bdata;
public:
B(int a,int b):A(a)
{bdata=b;}
void showdata()
{cout<<"\nadata="<<adata<<endl<<"bdata="<<bdata;}};
void main()
{B b(5,6);
```

```
b.showdata();
getch();}
```

//Removal of ambiguity in multiple inheritance

```
#include<iostream.h>
#include<conio.h>
class A
{public:
void show()
{cout<<"\nThis is Class A"<<endl;
}};
class B
{public:
void show()
{cout<<"This is class B"<<endl;}};
class C:public A,public B
{public:
void show()
{A::show();
B::show();}};
int main()
{C c;
c.show();
c.A::show();
c.B::show();
getch();
}
```

//Derived Class Destructor

```
#include<iostream.h>
#include<conio.h>
class A
{public:
~A()
{cout<<"\nClass A Destructor"<<endl;}};
class B: public A
{public:
~B(){cout<<"\nClass B Destructor"<<endl;}};
class C:public B
{public:
~C(){cout<<"\nClass C Destructor"<<endl;}};
void main()
{{C x;
}//destructor is called at this point
getch();}
```

//Aggregation(Containership)

```
#include<iostream.h>
#include<conio.h>
class Employee
{
int eid,sal;
public:
void getdata()
{cout<<"Enter id and salary of employee"<<endl;
cin>>eid>>sal;
}
void display()
{cout<<"Emp ID:"<<eid<<endl<<"Salary:"<<sal;}};
class company
{int cid;
char cname[20];
Employee e;
//Containership, Object of Employee class is included in Company class
public:
void getdata()
{cout<<"Enter id and name of the company:"<<endl;
cin>>cid>>cname;
e.getdata();
}
void display()
{cout<<"Comp ID:"<<cid<<endl<<"Comp Name:"<<cname;
e.display();
}};
void main()
{company c;
c.getdata();
cout<<"\n##########Company Details########\n"<<endl;
c.display();
  getch();}
```

//Hybrid Inheritance
//Virtual Base Class

```
 #include<iostream.h>
      #include<conio.h>
      class ClassA
      {     public:       int a;      };
      class ClassB : virtual public ClassA
      {    public:       int b;      };
      class ClassC : virtual public ClassA
      {    public:     int c;     };
```

```
class ClassD : public ClassB, public ClassC
      {    public:    int d;      };
      void main()
      {    ClassD obj;
  obj.a = 10;       //Statement 1
  obj.a = 100;      //Statement 2
         obj.b = 20; obj.c = 30;
         obj.d = 40;
         cout<< "\n A : "<< obj.a;              cout<< "\n B : "<< obj.b;
         cout<< "\n C : "<< obj.c;
         cout<< "\n D : "<< obj.d;       }

//Example of using friend function
#include <iostream.h>
#include<conio.h>
class B;
class A {
   private:      int numA;
   public:      A(): numA(12) { }
     // friend function declaration
     friend int add(A, B);};
class B {    private:
      int numB;
   public:      B(): numB(1) { }
      // friend function declaration
      friend int add(A,B);};
// Function add() is the friend function of classes A and B
// that accesses the member variables numA and numB
int add(A objectA, B objectB)
{   return (objectA.numA + objectB.numB);}
int main()
{    A objectA;
   B objectB;
   cout<<"Sum: "<< add(objectA, objectB);
   getch();return 0;}
```

Virtual Function, Polymorphism, and miscellaneous C++ Features

//Example of a virtual function

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class b
{public:
virtual void show()
{cout<<"\n  Showing base class....";}
```

```
void display()
{cout<<"\n  Displaying base class...." ;}
};
class d:public b
{public:
void display()
{cout<<"\n  Displaying derived class....";}
void show()
{cout<<"\n  Showing derived class....";}
};
```

Pure virtual function

```
#include <iostream>
using namespace std;
class Employee                  //  abstract base class
{    virtual int getSalary() = 0;    // pure virtual function};
class Developer : public Employee
{    int salary;
     public:
        Developer(int s)       {salary = s;    }
        int getSalary()        {return salary;}};
class Driver : public Employee
{    int salary;
     public:
        Driver(int t)    {salary = t;     }
        int getSalary()        {return salary;}};
int main()
{    Developer d1(5000);
     Driver d2(3000);
     int a, b;
     a = d1.getSalary();
     b = d2.getSalary();
     cout << "Salary of Developer : " << a << endl;
     cout << "Salary of Driver : " << b << endl;
     return 0;
}
```

//Virtual Base Class

```
#include<iostream.h>
#include<conio.h>
     class ClassA
     {      public:
            int a;       };
     class ClassB : virtual public ClassA
     {     public:
            int b;       };
     class ClassC : virtual public ClassA
     {     public:
            int c;      };

     class ClassD : public ClassB, public ClassC
     {     public:
            int d;       };
     void main()
```

```
    {  ClassD obj;
          obj.a = 10;        //Statement 1
          obj.a = 100;       //Statement 2
          obj.b = 20;
          obj.c = 30;
          obj.d = 40;
            cout<< "\n A : "<< obj.a;
          cout<< "\n B : "<< obj.b;
          cout<< "\n C : "<< obj.c;
          cout<< "\n D : "<< obj.d;      }
 Output :
          A : 100
          B: 20
          C : 30
          D : 40
```

//Overloading + operator using friend function

```
#include<iostream.h>
#include<conio.h>
class distance
{private:
   int feet, inches;
public:
   distance(){feet=inches=0;}
   distance(int f,int in){feet=f;inches=in;}
friend distance operator +(distance,distance);
int display()
{    cout<<"("<<feet<<","<<inches<<")"<<endl;
return 0;}};
distance operator +(distance x,distance y)
{    distance r;
   r.feet=x.feet+y.feet;
   r.inches=x.inches+y.inches;
   r.feet=r.feet+r.inches/12;
   r.inches=r.inches%12;
   return r;}
int main()
{    distance d1(5,6),d2(7,8),d3;
   d3=d1+d2;//d1.operator +(d2);
   cout<<"d1=";
   d1.display();
   cout<<"d2=";
   d2.display();
   cout<<"d3=";
   d3.display();
   getch();}
```

I

/Static Function

```
#include <iostream.h>
#include <conio.h>
class Demo
{private:
        //static data members
                static int X;
                static int Y;
        public:
        //static member function
        static void  Print()
{cout <<"Value of X: " << X << endl;
cout <<"Value of Y: " << Y << endl;
}};
int Demo::X=10;
int Demo::Y=20;
//static data members initializations
void main()
{Demo OB;
//accessing class name with object name
cout<<"Printing through object name:"<<endl;
OB.Print();
//accessing class name with class name
cout<<"Printing through class name:"<<endl;
Demo::Print();
getch();}
```

//Example of using friend function

```
#include <iostream.h>
#include<conio.h>
class B;
class A {
   private:
     int numA;
   public:
     A(): numA(12) { }
     // friend function declaration
     friend int add(A, B);};
class B {    private:
      int numB;
   public:
      B(): numB(1) { }
      // friend function declaration
      friend int add(A,B);};
// Function add() is the friend function of classes A and B
// that accesses the member variables numA and numB
int add(A objectA, B objectB)
{   return (objectA.numA + objectB.numB);}
int main()
{    A objectA;
```

```
    B objectB;
    cout<<"Sum: "<< add(objectA, objectB);
    getch();return 0;}
```

Use of 'this' when local variables and class data members are same (while defining a member function)

```
#include <iostream.h>
#include <conio.h>
class Number
{    private:
        int a;
     public:
        void get_a(int a)
        {this->a=a;}
        void put_a()   {cout<<"a= "<<a<<endl;}};
int main()
{    Number N;       N.get_a(36);
     N.put_a();getch();
     return 0;}
```

Use of 'this' also for returning the invoking object

```
#include <iostream.h>
#include <conio.h>
#include<string.h>
class person{
char *name;int age;
public:
void setdata(char *name,int age)
{strcpy(this->name,name);//name conflict resolution
this->age=age;}
void display()
{cout<<"\nName:"<<this->name<<endl;
cout<<"Age:"<<this->age<<endl;}
person isElder(person p)
{if(age>p.age)
return *this;//returning invoking object
else
return p;}};
void main()
{person p,p1,p2;
p1.setdata("Aryan",12);
p2.setdata("Binek",22);
p=p1.isElder(p2);
cout<<"Elder one is:"<<endl;
p.display();getch();}
```

Function Templates and Exception Handling

//Finding maximum of two values by using template function

```
#include<iostream.h>
#include<conio.h>
template<class T>
T max(T a,T b)
{T result;
result=(a>b)?a:b;
return result;}
void main()
{int a=5,b=6,k;
float l=10,m=5,n;
char x='a',y='b',z;
k=max(a,b);
n=max(l,m);
z=max(x,y);
cout<<"\nLarger of integers:"<<k;
cout<<"\nLarger of floats:"<<n;
cout<<"\nLarger of characters:"<<z;
getch();
}
```

//Finding minimum of two values of different types by using template function

```
#include<iostream.h>
#include<conio.h>
template<class T,class U>
T min(T a,U b)
{T result;
result=(a<b)?a:b;
return result;}
void main()
{float x=5,r;
int y=3;
r=min(x,y);
cout<<"\nSmaller:"<<r;
getch();
}
```

//Finding greater number using class template

```
#include<iostream.h>
#include<conio.h>
template<class T>
class mypair
{T a,b;
public:
```

```
mypair(T first,T second)
{a=first;b=second;}
T max()
{T r;
r=(a>b)?a:b;
return r;
}};
void main()
{mypair<int> ob1(200,500);
cout<<"\nLarger integer:"<<ob1.max();
mypair<double> ob2(9.8,3.6);
cout<<"\nLarger double:"<<ob2.max();
getch();}
```

//program to add two numbers using function template.

```
//function template
#include<iostream.h>
#include<conio.h>
template<class t1,class t2>
void sum(t1 a,t2 b) // defining template function
{
   cout<<"Sum="<<a+b<<endl;
}

void main()
{
   int a,b;
   float x,y;
   cout<<"Enter two integer data: ";
   cin>>a>>b;
   cout<<"Enter two float data: ";
   cin>>x>>y;
   sum(a,b); // adding two integer type data
   sum(x,y); // adding two float type data
   sum(a,x); // adding a float and integer type data
   getch(); }
```

Exception Handling

Simple program for exception handling using try, throw and catch

```
#include<iostream>
using namespace std;
int main()
{   try {
      throw 6;   }
   catch (int a) {
      cout << "An exception occurred!" << endl;
      cout << "Exception number is: " << a << endl;
```

```
}}
```

Below program contains single catch statement to handle errors.

```
#include <iostream>
#include<conio.h>
using namespace std;
int main()
{int a=10,b=0,c;
try //try block activates exception handling
{if(b==0)
throw "Division by zero not possible";//throws exception
c=a/b;}
catch(char* ex)//catches exception
{cout<<ex;
}
getch();
return 0;
}
```

File handling (6 Hrs.)

Program to create a binary file 'student.dat' using class

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class student{
int roll;
char name[30];
char add[60];
public:
void readdata();
};
void student::readdata()
{
cout<<"\nEnter Roll:";
cin>>roll;
cout<<"\nStudent Name:";
cin>>name;
cout<<"\nEnter Address:";
cin>>add;}
void main()
{char ch;
student s;
ofstream fout;
fout.open("student.dat",ios::app);
do{
s.readdata();
fout.write((char *)&s,sizeof(student));
cout<<"Do you want to add more ?(Y/N):";
cin>>ch;}
while(ch=='y'||ch=='Y');
fout.close();
}
```

Program to read a binary file 'student.dat' display records on monitor.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class student{
```

```
int roll;
char name[30];
char add[60];
public:
void readdata();
};
void student::readdata()
{cout<<"\nRoll:"<<roll;
cout<<"\nName:"<<name;
cout<<"\nAddress:"<<add;
}
void main()
{
student s;
ifstream fin;
fin.open("student.dat",ios::binary|ios::in);
fin.read((char *)&s,sizeof(student));
while(fin)
{ s.readdata();
fin.read((char *)&s,sizeof(student));
 }
fin.close();
getch();
}
```

Program to create a binary file ‘student.dat’ using structure.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
struct student
{
char name[15];
float percent;
};
void main()
{
ofstream fout;  // f out is output file stream it will open file in write mode
char ch;
fout.open("student.dat",ios::app|ios::binary);//student.dat will be opened in
// binary Mode
clrscr();
 student s;   //s is a variable of type student that contains name & percent
do
{       // inputting data to record  s
cout<<"\n enter name of student";
cin>>s.name;
```

```
cout<<"\n enter persentage";
cin>>s.percent;
//Writing contents of s to file student.dat
fout.write ((char * )&s, sizeof(s));
cout<<"\n more record y/n";
cin>>ch;
}while(ch=='y'||ch=='Y');
fout.close();
}
```

Program to read a binary file 'student.dat' display records on monitor.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
struct student
{
char name[15];
float percent;
};
void main()
{
ifstream fin; //fin is an input file stream that can open a file in read mode
student s;    // s is a record of type student
fin.open("student.dat",ios::in | ios:: binary); // opening student.dat in binary mode
fin.read((char *) &s, sizeof(student)); //read a record from file 'student.dat' invariable s
while(fin) // file will read until end of file does not come.
{
        //Displaying the content of  s (reord) read from the student.dat
cout<<endl<<s.name;
cout<<"\n has the percent:"<<s.percent;
fin.read((char *)&s, sizeof(student)); // reading the next record
}
fin.close();
getch();
}
```

Reading the contents of a file using getline() method and detecting the end of file using EOF() member function.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
char str[100];
ifstream fin;
fin.open("abc.txt");
```

```
while(!fin.eof())
{fin.getline(str,99);
cout<<str;
}
fin.close();
getch();
}
```

Writing the text in a file

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
ofstream fout;
fout.open("abc1.txt");
fout<<"This is Texas College";
fout<<"\nMitrapark, Kathmandu";
fout.close();
getch();
}
```

Reading the contents of a file using getline() method and detecting the end of file using filestream object.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
char str[100];
ifstream fin;
fin.open("abc.txt");
while(fin)
{fin.getline(str,99);
cout<<str;
}
fin.close();
getch();
}
```

Sample Project for Students' Information System

A project on Students' Information system which is menu base and has the features to add new records of students, find the individual record, update the record and delete the record.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class student{
int roll;
char name[30];
char add[60];
public:
void input()
{
cout<<"\nEnter Roll:";
cin>>roll; cin.sync();
cout<<"\nStudent Name:";
cin.getline(name,30);
cout<<"\nEnter Address:";
cin.getline(add,60);}
void display()
{cout<<"\nRoll:"<<roll;
cout<<"\nName:"<<name;
cout<<"\nAddress:"<<add;
}
int getrno()
{return roll;}
};
void updaterecord()
{student s;fstream fin;int key;int found=0,pos;
fin.open("student.dat",ios::binary|ios::in|ios::out);
fin.read((char *)&s,sizeof(student));
cout<<"Enter the Roll No. to search record for update:";cin>>key;
while(fin)
{ if(key==s.getrno())
{cout<<"\nExisting Record:\n";s.display();found=1;pos=fin.tellg()-sizeof(s);cout<<"\nPos:"<<pos;
fin.seekp(pos,ios::beg);
cout<<"\nEdit the record:\n";
s.input();
fin.write((char *)&s,sizeof(student));
}
fin.read((char *)&s,sizeof(student));
 }if(found==1)
 cout<<"\nSearch Successful....\n";
 else
```

```
 cout<<"\nRecord not found...\n";
fin.close();
}
void delrecord()
{fstream file("student.dat", ios::in|ios::binary);int r;
fstream newfile("newstu.dat",ios::out|ios::binary);
student s;
cout<<"\n enter the rollno no of student whose record to be deleted";
cin>>r;
file.read((char *)&s,sizeof(s));
while(file)
{if (r!=s.getrno())
{newfile.write((char *)&s,sizeof(s));}
file.read((char *)&s,sizeof(s));}
file.close();
newfile.close();
fstream file2("student.dat", ios::out|ios::binary);
fstream newfile2("newstu.dat",ios::in|ios::binary);
newfile2.read((char *)&s,sizeof(s));
while(newfile2)
{file2.write((char *)&s,sizeof(s));
newfile2.read((char *)&s,sizeof(s));}
file2.close();
newfile2.close();
}

void findrecord()
{student s;ifstream fin;int key;int found=0;
fin.open("student.dat",ios::binary|ios::in);
fin.read((char *)&s,sizeof(student));
cout<<"Enter the Roll No. to search:";cin>>key;
while(fin)
{ if(key==s.getrno())
{s.display();found=1;}
fin.read((char *)&s,sizeof(student));
 }if(found==1)
 cout<<"\nSearch Successful....\n";
 else
 cout<<"\nRecord not found...\n";
fin.close();
}
void main()
{int ch;clrscr();
do{
cout<<"\n***Menu***\n1.Input Record\n2.Display Record\n3.Find Record\n4.Update
     Record\n5.Delete Record\n";
```

```
cout<<"Enter Your Choice:";cin>>ch;
switch(ch){
case 1:
{student s;s.input();
ofstream fout;
fout.open("student.dat",ios::app);
fout.write((char *)&s,sizeof(student));
fout.close();break;  }
case 2:
{student s;ifstream fin;
fin.open("student.dat",ios::binary|ios::in);
fin.read((char *)&s,sizeof(student));
while(fin)
{ s.display();
fin.read((char *)&s,sizeof(student));
 }
fin.close();break;
}
case 3: findrecord();break;
case 4: updaterecord();break;
case 5: delrecord();break;
}}
while(ch<6);
 }
```