Experiment-6

1. Single Inheritance

```
# Include <iostream>
using namespace std;
 class person
{
      protected:
      String name;
       int age;
    };
    class Student : Protected Person
  {
       private:
        int roll;
         public:
    void accept()
    {
      cout << " Enter name";
      cin >> name;
      cout << " Enter age";
      cin >> age;
      cout << "Enter roll no.";
      cin >> roll;
    }
        void display()
  {
      cout << " Name" << name;
      cout << " Age" << age
      cout << " Roll no." << roll;
  }
};
```

```
int main ()
{
      Student S1;
      S1. accept();
      S1. display();
       return 0;
}
```

O/p. Enter Name Dhwani
Enter Age 17
Enter Roll no 37

Name Dhwani
Age 17
Roll no 37

2) Multiple Inheritance

```
# Include <iostream>
using namespace std;
class academic
 {
      protected:
      String name;
       int marks;
    };
    class sports
    {
        protected:
          int score;
    };
```

Class Student : protected academic, protected Sports
{
int total;
public :
void accept()
{
cout << " Enter name";
cin >> name;
cout << " Enter marks";
cin >> marks;
cout << "Enter score";
cin >> score;
total = marks + score;
}
void display()
{
cout << "Name" << name;
cout << " Marks" << marks;
cout << "Score" << score;
cout << " Total" ;
}
};
int main()
{
student s;
s.accept();
s.display();
return 0;
}

3 Multilevel:

#include <iostream>
using namespace std;
class vehical
{
public:
string model;
string brand;
};
class car: public vehical
{
protected:
int type;
};
class electriccar : public car
{
public:
int batterycapacity;
void accept()
{
cout << "Enter Model";
cin >> model;
cout << " Enter brand;
cin >> brand;
cout << "Enter Type (1 for Sedan, 2 for SUV)";
cin >> type;
cout << " Enter Battery capacity (in KWH)";
cin >> battery capacity;
}

```
Void display ()
{
    cout << " Model" << model;
    cout << " Brand" << brand;
    cout << " Type" << (type == 1?"sedan":
                        "suv");
    cout << " Battery capacity " <<
    c;                  battery_capacity
                        << " KWH";

  }
};

    int main ()
{
    electriccar C;
    C. accept ();
    C. display ();
    return 0;
}
```


4. Hierarchial

```
#include <std iostream>
  using namespace std;
    class employee
  {
      protected:
      string name;
       int id;
  };

  class manager : public employee
    {
        private:
         string dep;
    };

class Developer : public employee
  {
      string : public:
        String lang;
        String dep;
        Void accept()
    {
        cout << "Enter emp name";
        cin >> name;
        cout << " Enter emp id";
        cin >> id;
        cout << "Enter programming
                Language";
        cin >> lang;
        cout << " Enter department"
    }  cin >> dep;
```

```
void display ()
{
    cout << " Name" << name;
    cout << " ID" << id;
    cout << " Programming language"
                                << lang;
    cout << " Department" << dep;
  }
};

    int main ()
  {
      Developer d;
      d. accept();
      d. display();
      return 0;
  }
```

5. hybrid

```
#include <iostream>
 using namespace std;
  class teacher
  {
      protected:
       string name;
       string age;
       public.
        void accept()
    {
      cout << " Enter Teacher Name";
      cin>> name;
      cout<<" Enter Teacher Age";
       cin>> age;
  }
      void display ()
    {
      cout <<" Teacher Name"
          << name;
      cout <<" Teacher age"
          << age;
    }
  };

class Student : public Teacher
{
    protected:
      int roll;
      string sname;
```

```
public:
  void saccept()
  {
    cout <<" Enter Student name";
    cin>> sname;
    cout <<" Enter roll no.";
    cin>> roll;
  }
    void sdisplay()
    {
      cout <<" Student Name"
          << sname;
      cout <<" Roll number"
          << roll;
    }
};

class marks : public Student
{
  protected:
  int m1, m2, m3;
  public:
  void maccept()
  {
    cout <<" Enter marks";
    cin >> m1 >> m2 >> m3;
  }

    void mdisplay()
    {
      cout <<" Marks in sub.1"
          << m1;
      cout <<" Marks in sub.2";
      cin >> m2;
      cout <<" Marks in sub.3";
      cin >> m3;
    }
};

class academics: public marks
{
    public:
    int totalmarks()
    {
      return m1 + m2 + m3;
    }
};

int main()
{
  academics a;
  a.accept();
  a.saccept();
  a.maccept();
  a.display();
  a.sdisplay();
  a.mdisplay();

  INT total = a.totalmarks();
  cout <<" Total marks" <<total;
  return 0;
}
```

12/11

Experiment-7

a) WAP using functions overloading to calculate the area of a laboratory and area of classroom.

```
#include <iostream>
  using namespace std;
   class xyz
{
      public:
     int length, breadth, side, area;

      void calculateArea (int len, int br)
      {
          length = len;
          breadth = br;
          area = length * breadth;
      }
          void calculateArea (int s)
          {
              side = s;
              area = area * side;
          }
};
    int main ()
    {
        xyz a;
    cout << " Enter length and breadth";
     cin >> a. length >> a. breadth;
  a. calculateArea (a.length,
                    a.breadth);
```



```
cout << " Area of rectangle " << a.area;

xyz b;
cout << " Enter side of Square';
 cin >> b.side;
 b.calculateArea (b.side);
 cout << " Area of Square " << b.area;
  return 0;
}
```

o/p:

Enter length and breadth
3
2
Area of rectangle 6
Enter side of Square
5
Area of Square 25

b) WAP using function overloading to calculate the sum of 5 float values and sum of 10 integer values.

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

class sum
{
    public:
    float floatsum;
    int intsum;
    void cal(float a, float b)
    {
        floatsum = a+b;
    }
    void cal(int a, int b)
    {
        int sum = a+b;
    }
};
    int main()
{
        sum obj;
        float f1, f2;
cout << "Enter 2 float values";
cin >> f1 >> f2;
obj.cal(f1,f2);
cout << "Sum of float values"
<< obj.floatsum;
int i1, i2;
cout << "Enter 2 integer";
cin >> i1 >> i2;
obj.cal(i1, i2);
cout << "Sum of integer values"
<< obj.intsum;
return 0;
}
```

c. WAP to implement Unary Operator when used to with the object so that the numeric data members of the class is negated.

```
#include <iostream>
using namespace std;
class numb
{
    public:
    void accept()
    {
        cout << "Enter a number";
        cin >> value;
    }
    void display()
    {
        cout << "Value" << value;
    }
    void operator-()
    {
        value = -value;
    }
};

int main()
{
    numb obj;
    obj.accept();
    -obj;
    obj.display();
    return 0;
}
```

d) WAP to implement the Unary ++ operator (for pre increment and post increment) when used with object so that the numeric data members of the class is incremented.

```
#include <iostream>
using namespace std;
class ab
{
    public:
    int n;
    void accept()
    {
        cout << "Enter number";
        cin >> n;
    }
    void display()
    {
        cout << "Number" << n;
    }
    void operator++()
    {
        ++n;
    }
    void operator++(int)
    {
        n++;
    }
};
```

```
int main()
{
    ab obj;
    obj acc;
    ++ obj;
    obj++;
    obj display();
    return 0;
}
```

12/11

Experiment-8

a) WAP to overload the '+' operator so that two strings can be concatenated.

```
#include <iostream>
#include <string>
using namespace std;
class abc
{
    public:
    string s;
    void accept()
    {
        cout << "Enter string";
        cin >> s;
    }
    abc operator + (abc s1)
    {
        abc temp;
        temp.s = s + s1.s;
        return temp;
    }
    void display()
    {
        cout << "concated string" << s;
    }
};
```

```
int main()
{
    abc st1, st2, r;
    st1.accept();
    st2.accept();
    r = st1+st2;
    r.display();
    return 0;
}
```

o/p:

Enter String: Dhwani
Enter string: Zavery
Concated String: Dhwani Zavery

b) WAP to create a base class Ilogin having data members name and password Declare accept() function virtual. Derive Emaillogin and Membershiplogin classes from Ilogin. Display Email login details and memberships login details of the employees.

```
#include<iostream>
using namespace std;
class Ilogin
{
    protected:
    String name, password;
    public:
     void accept()
    {
      cout << "Name";
      cin >> name;
      cout << " Password";
      cin >> password;
    }
};
class Emaillogin: virtual public Ilogin
{
    public:
     void ShowEmail()
    {
        cout << name << " " << password;
    }
};
```

```
class Membershiplogin : virtual public Ilogin
{
    public:
    void ShowMembership()
    {
      cout << name << " " <<password;
    }
};

class Employee : public Emaillogin,
                 public Membershiplogin
{
    void input()
    {
        accept();
    }

    void display()
{
    ShowEmail();
    ShowMembership();
}
};

int main()
{
    Employee e;
    e.input();
    e.display();
    return 0;
}
```


Experiment - 9

```
# include <iostream>
# include <istream>
# include <string>
# include <sstring>
  using namespace std;
   int main()
{
      ifstream fin;
       ofstream fout;
      fin.open("source.text");
      fout.open("destination.txt");

      if (!fin)
    {
      cout << " Error";
      return 1;
    }
      char ch;
    while (finget(ch))
     {
        fout.put(ch);
     }
      fin.close();
      fout.close();
     fin.open("Source.txt");
      String word;
     int count=0
      while(fin >> word)
     {
        count ++;
     }
```

```
cout << "The word count is " << count;
fin.close();
fin.open("source.txt");
String target;
  int c=0;
  cout << " Enter Target";
  cin>> target;
  while(fin>>word)
   {
      if(word == target)
    {
        c++;
      }
    }
cout << " Target found at" << c;
}
```

Q)
12/11

Experiment 1

a)
```
# include <iostream>
using namespace std;
template <class T>
T sum (T a[], int n)
{
    T sum=0;
for(int i=0; i<n; i++)
{
    sum+ =a[i];
 }
    return sum;
}
    int main()
  {
      int n=5;
      int inta[n];
      float floata[n];
      double doublea[n];
      cout << " Enter 5 numbers";
    for (int i=0; i<n; i++)
{
    cin>> inta[i];
 }
    cout << " Enter 5 integer elemen
    for(int i=0; i<n; i++)
{
    cin >> floata[i];
}
    cout << " Enter 5 double numbers";
    for(int i=0; i<n; i++)
{   cout doublea[i];
 }
```

```
cout<<" Sum of integer nos"<<sum(int a, n);
cout<<"  ——,  float  —"— <<sum(float a, n);
cout<<" ——"— double "— <<sum(double a, n);
}
```

b)
```
# include <iostream>
# include <string>
 using namespace std;
 template <class T>
  T square (T n)
  {
     return n*n;
  }
    template <>
   string square <string> (string s)
  {
     return S+S;
  }
     int main ()
  {
       int num;
       String str;
  cout << " Enter a integer");
    cin>> cin>> num;
  cout<<" Enter a String ";
    cin >> str;

  cout <<" Square of integer " << num <<
  cout << " Square of string" << str << "=2"
                                square (str);

  }
```


c)
```
#include <iostream>
#include <cmath>
 using namespace std;
 Template <class T>
  class cal
  { public:
      T a, b;
      void accept()
    {
      cout << " Enter 2 nos";
      cin >> a >> b;
    }
      void add ()
    {
      cout << " Addition " << a+b;
    }
      void sub()
    {
      cout<<" subtraction" << a-b;
    }
      void mul ()
    {
      cout << " Multiplication " << a*b;
    }
      void div()
    {
      cout << " Division" << a/b;
    }
};
```

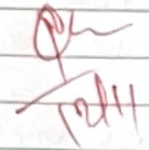
```
int main(){
cal <double> n;
   int ch;
   n. accept();
   while (1)
  {
    cout << " 1. Addition \n 2. Subtraction \n
              3. Multiplication \n
              4. Division 5. Exit";
    cout << " Enter choice";
    cin >> ch;
    switch (ch)
  {
     case 1: n. add();
          break;
  case 2: sub();
          break;
  case 3: mul();
          break;
  case 4: div();
          break;
  case 5: exit(0);
  default << " Wrong choice";
          break;
  }
  }
  }
```

12/11

Experiment - 11

1)
```
#include <iostream>
#include <vector>
#include <ctype>
 using namespace std;
     int  main()
{
       vector <int> vec[5];
        int i;
       cout << " Enter 5 elements";
        for(i=0; i<5; i++)
      {
          cout << vec[i];
      }
       cout<< " Modified elements";
      for(i=0; i<5; i++)
    {
       vec[i] = vec[i] + i*2;
    }
      for (i=0; i<5 ; i++)
     {
        cout << vec[i] << " ";
     }
          int scalar;
          cout << " Enter a val";
          cin>> scalar;
          cout << " after Multiplica
      for (i=0; i<5; i++)
     {
          vec[i] = vec[i] * scalar;
```

```
for (i=0; i<5; i++)
{
    cout << vec[i] << "   ";
}
    cout << endl;
}
```



Experiment - 12

a)
```
#include <iostream>
#include <stack>
#include <ctype>
using namespace std;
    int main()
{
        stack <int> v;
        v.push(1);
        v.push(2);
        v.push(3);
        v.push(4);
        v.push(5);
    if (v.empty())
    {
        cout << "stack is empty";
    }
        else
    {
        cout << "stack is not empty";
    }
    cout << "Size " << v.size();
    cout << "topmost" << v.top();
     cout << "stack Empty";
    while(v.empty())
    {
        cout << v.top() << " ";
                    v.pop;
    }
        cout << "size after popping "
                    << v.size();
    }
```

b)
```
#include <iostream>
#include <queue>
# include <ctype>
    using namespace std;
        int main()
{
        queue <int> v;
          v. push (11);
          v. push (22);
          v. push(33);
          v. push (44);
          v. push (55);
        if( v. empty ())
{
    cout <<" Queue Empty";
}
        else
        {
            cout <<" Queue not
                    empty",
        }
        cout <<" Size " <<v.size();
        cout <<" front " <<v. front();
        cout <<" Back" << v. back;
        cout <<" Queue";
        while( ! v. empty())
{
    cout << v.front ()<<"  ";v.pop()
}
    cout << v.size();
    }
```

@u
12/11