Enter bank account balance : 30000

Auditing ..... Bank account balance : 30000

12/8/25

Experiment 5

* Write a program to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

. Default Constructor :-

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

class addition
{
   int n;
   int sum;
public:
   addition ()
{
    n = 10;
    sum = 0;
for (int i=1; i<=n; i++)
{
  sum += i;
}
}
void display ()
{
cout << "Sum of first " << n << "numbers:"
                          << sum;
}
};
```

```
int main()
{
addition a;
a.display();
return 0;
}
```

- Parameterized constructor

```
#include <iostream>
using namespace std;
class addion
{
 int n, sum;
 public:
 addtion (int no.)
{
 n = n^e
 sum = 0
 for (int i = 1; ; i <= n; ; i++)
{
 sum + = i;
 }
 }
void display ()
{
 cout << "Sum of first" << n << "number
 << sum;
 }
};
int main () {
 addition . a (5);
 a . display ();

 return 0;
 }
```

Copy constructor

```
#include <iostream>
using namespace std;
class sum
{
  int n;
int s=0
public:
 sum()
{
  n=4;
}
sum(sum &obj)
{
 n=obj.n;
int i;
for(i=0;i<=n;i++)
{
 s=s+i;
}
cout<<"sum:"<<s;
}
};
  int main()
{
 sum s1;
sum s2(s1);
 return 0;

}
```

52) Write a program to declare a class student having data members name & percentage. Write a constructor to initialize these data members. Accept and display data for student.

. Default constructor

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

Class student
{
  int percentage;
  string name;
public:
   student()
  {
  name = "Arsalaan";
  percentage = 92;
}
  void display()
  {
  Cout << "Name of student:" << name << endl;
Cout << "Percentage of student:" << percentage
  };
int main()
  {
    student s1
    s1. display;
    return 0;
  }
```

1. Parameterized Constructor

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

class student
{
  float peretage;
  string name;
  public:
   student (string n, float p)
   {
     name = n;
     percentage = p;
   }
   void display ()
   {
  cout << "Name of student" << name << endl
  cout << "Percentage of student: " << percentage;
   }
  };
   int main ()
  {
   student s, ("Areaderm", 92);
   s. display ();
   return 0;

  }
```

classmate
Date
Page

. Copy Constructor.

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

class student
{
  private:
    string name;
    float percentage;
  public:
    student (string n, float p)
    {
      name = n;
      percentage = p;
    }
    void display()
    {
cout << "Name :" << name << endl;
cout << "percentage :" << percentage << "%." << endl;
    }
};
  int main()
  {
    student s1("Arsalaan", 92);
    student s2 = s1;
    s2 display();
    return 0;
  }
```

3) Define a class 'College' members variable as roll_no, name course, with using constructor with default value as computer Engineering for course create this data for two objects of class and display the data.

=> C- code :-

```
#include <iostream>
using namespace std;
class College
{
    int roll-no;
    string name;
    string course;
public:
College (int r, string n, string c = " compulsir
                    Engineering")
{
    roll-no = r;
    name = n;
    course = c;
}
void display ()
{ cout << "Roll no :" << roll-no << "Name:" <<
  name << "Coure:" << course << endl;
}};
int main() {
College S1 (1, "Arsalaan");
College S2 (2, "Sahil", "IT");
S1. display ();
S2. display ();
return 0; }
```

4 Write a program to demonstrate constructor overloading.

⇒ 1. code

```
#include <iostream>
using namespace std;
class student
{  int roll-no;
   string name;
 public:
   student()
 { roll-no = 0;
   name = "Unknown"; }
 student(int r)
 {
   roll-no = r;
   name = "Unknown"; }
 student(int r, string n)
 {  roll-no = r;
    name = n;  }
 void display()
 { cout << "Roll No: " << roll-no << ", Name: " <<
       name << endl; }
};
int main()
{
  Student s1;
 student s2(101);
 student s3(102, "Arsalwan");
 s1.display();
  return 0;

}
```

Experiment - 6

Write a C++ program to implement inheritance :-

1) Single inheritence

Create a base class called person with attributes name and age. Derive a class student from person that adds an attribute roll number. Write functions to display all details of the student.

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

Class person {
public:
string name;
int age;
void set person (string n, int a) {
string name;
int age;
public:
void accept person details () {
Cout << "enter name";
Cin >> name;
Cout << "enter age";
Cin >> age;
}
```

```
Void display person details () {
cout << "Name :" << name << endl;
cout << "age :" << age << endl;
    }
};

class student : public person {
 private
    int roll number;
public:
 Void accept student details () {
 accept person details ();
 cout << "enter roll number";
 cin >> roll number;
 }
 void display student details (){
  display person details ();
cout << "Roll Number :" << rollnumber " " << endl;
    }
};
int main () {
    Student S;
S. accept student details ();
 cout << "\n student detail";
  S. display student details ();

return 0;

 }
```

Output :-

- enter name = Sakshi
enter age = 17
enter roll no. = 13

Details of student :-
name : adam
age : 20
roll : 20

Q2] Multi inheritance :-
Create two base classes academic and sports.

- academic class contains marks of a student

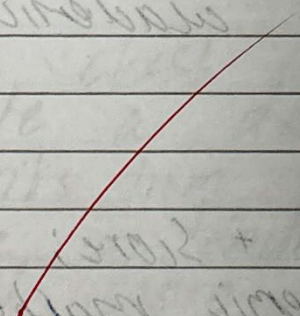
- Sports class contains sports score.
Create a derived class result that inherits from both academic and sports while a function to calculate the total score and display details

⇒
```
#include <iostream>
using namespace std
class academic {
public:
        int marks;
        void getmarks () {
cout << "enter academic marks";
cin >> marks;
}};
class sports {
public:
  int score;
  void get score () {
cout << "enter sports score:";
cin >> score;
}
};
class result : public academic;
public sports {
public:
  void display () {
    int total = marks + score;
    cout << "\n academic marks" <<
            marks;
cout << "\n sports score:" << score;
cout << "\n total score:" << total << endl;
}};
  int main () {
     result r;
     r.get marks ();
     r.get score ();
     r.display ();
```

```
return 0;
}
```

Output :-

```
enter academic marks : 90
enter sports score : 95

academic marks : 90
sports score : 95
total score : 185
```



3] Creat a class vehicle with attributes like brand and model. Derive a class from car from vehicl which add an attribute type Further derive a class Electric car from car which add battery capacity. Write function to display all the details

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

class vehicle {
public:
    string brand, model;
};

class car : public vehicle {
public:
    string type;
};

class Electricar : public car {
public:
    int battery capacity;
void input ()
cout << "Enter brand";
cin >> brand;
cout << "Enter model";
cin >> model;
cout << "Enter type:";
cin >> type;
cout << "Enter battery capacity (KWh):";
cin >> battery capacity;
```

```
};

Void display (){
 Cout << "\n... Electric Car Details ...\n";
 Cout << "Brand:" << brand << endl;
 Cout << "Model:" << type << endl;
 Cout << "Type:" << type << endl;
 Cout << "Battery Capacity:" << batter capacity <<
                                          endl;
 }
   };

   int main (){
     Electriccar e;
     e.input ();
     e.display ();
     return 0;

   }
```

Output:-

```
Enter brand : BMW
Enter model : X7
Enter type : SUV
Enter battery capacity : 50000

... Electric Car Details ...
Brand: BMW
Model : X7
Type : SUV
Battery capacity : 50000 (Kwh)
```

4] Creat a base class Employee with attributes emp ID and name. Derive two classes manager and Developer from Employee Manger has attribute department and developer has an attribute programming language. Write function to display details for both

⇒
```
#include <iostream>
using namespace std;
class Employee
{
public:
int empID;
string name;
void setData(int id, string n)
{
    empID = id;
    name = n;
}
void display()
{
cout << "Employee I.D:" << empID << endl;
cout << "Name:" << name << endl;
}
};
class Manager : public Employee
{
public:
string department
void setDepartment(string dept)
```

```
{
    department = dept;
}
void display ()
{
    Employee :: display ();
    cout << "Department: " << department << endl;
}
};
class Developer : public Employee
{
    public:
    string programming language;
    void set Programming language (string lang)
{
    progamming language = lang;
}
    void display ()
{
    Employee:: display ();
    cout << "Progamming language: " << progamming language
        << endl;
}
};
int main ()
{
    Manager m;
    m.set Data (101, "Arsalaan");
    m.set Department ("HR")
    Developer d;
    d set Data (102, "Sohil");
```

d.set Programming language ("C++")

cout << " Manger Details : " << endl;
m.display();
cout << "\n Developer Details : " << endl;
d.display();
return 0;
}

⇒ Combine multilevel and multiple inheritance
Creat a base class Person with attributes
name and age. Derive class student from
Person. Create two classes sports and academics
Derive class Result from student as
sports

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

class Person
{ public:
   string name;
   int age;
   void setPerson (string n, int a)
{
    name = n;
    age = a;
}
  void display Person()
{
 cout << "Name: " << name << endl;
 cout << "Age: " << age << endl;
}
};
class Studat : public Person
{
public:
    int studentID;
    void setStudat (int ID)
{
```

```
    studentID = id;
}
void displaystudent ()
{
 cout << " Student ID : " << studentID << endl;
  }
  };
  class sports
  {
   public :
    int sportsscore;
   void setsports score (int score)
   {
     sport score = score;
   }
   void display sport ()
{
  cout << "sports score :" sportsscore << endl;
   }
   };
class Result : public student public sports
{
   Public :
   int marks;
   void set marks (int m)
{
  marks = m;
  }
  int calculate Total ()
{
  return marks + sportscore;
   }
```

```
void displayResult()
{
    displayPerson();
    displayStudent();
    displaySports();

cout << "Marks: " << marks << endl;
cout << "Total = (Marks + Sports Score): "
     << calculateTotal() << endl;
}
};
int main()
{
    Result r;
r.setPerson("John", 20);
r.setStudent(101);
r.setSportsScore(40)
r.setMarks(85);
r.displayResult();
return 0;
}
```

17/10

Experiment 7

Write a program to demonstrate compile time polymorphism (Function overloading & operator overloading - unary)

a) WAP using function overloading to calculate area of laboratory (rectangle) and area of classroom (square.

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

class Area
{
  public:
  float calculate (float length, float breadth)
  {
     return length * breadth;
  }
  float calculate (float side) {
     return side * side;
  }
  };
int main() {
Area a;
float length, breadth, side;
cout "Enter length, breadth of laboratory:";
cin >> length >> breadth;
cout << "Area of laboratory (rectangle):";
```

```
a.calculate (length, breadth);
                Enter side
Cout << " \n of classroom:";
Cin >> side;
Cout << "Area of classroom (Square):";
    a.calculate (side);

return 0;
}
```

Output:-

```
Enter length, breadth for laboratory
5       4
Area of laboratory (rectangle): 20
Enter side of classroom : 2
Area of classroom (square) : 4
```

82] WAP using function overloading to calculate sum of 5 float values & 10 integer values.

```
#include <iostream>
using namespace std;
class sumcalculator {
public:
float sum (float a, float b, float c, float d,
           float e)
{
    return a+b+c+d+e;
}
int sum (int a, int b, int c, int d, int e,
         int f, int g, int h, int i, int j){

  return a+b+c+d+e+f+g+h+i+j;

}
};
int main() {
  sum_calculator s;
float fsum = s.sum(1.1f*2.2f*3.3f*5.5f*4.4f);
int isum = s.sum(1,2,3,4,5,6,7,8,9,10);

cout << "sum of 5 float numbers:" << fsum;
cout << "sum of 10 integer nos:" << isum;

return 0; }
```

Output

Sum of 5 float numbers : 16.5
Sum of 10 integers nos : 55

Q3] WAP to implement unary (-) operator when used with the objects so that numeric data members of class is negated.

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

class number
{
int a;
public:
void accept(){
cout << "a : " << endl;
cin >> a;

void display(){
cout << "a = " << a << endl;
}
void operator -()
{
   a = -a;
}}
};
```

```
int main () {
    number n1;
    n1.accept();
    -n1;
    n1.display();
    return 0;
}
```

Output :-

a: 5
a = -5

Q4] Write a program to implement Unary (++) operator (for pre & post increment) when used with object so that numeric d... member of class is incremented.

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

class Number {
private:
Number (int v=0) : value (v) { }

Number & operator ++ () {
    value ++;
    return * this;
}
Number operator ++ (int)
Number temp = * this;
  value ++;
  return temp;
}
void display () {

  cout << value << endl;
}
};
  int main () {

Number n(5);
```

```
cout << "Original: ";
n.display();

cout << "After pre-increment (++n): ";
++n;
n.display();

cout << "After post-increment (n++): ";
n++;
n.display();

return 0;
}
```

Output :-

```
Original : 5
After pre-increment (++n) : 6
After post-increment (n++) : 7
```

17/10

Experiment 8

Q1] WAP to overload the '+' operator so that 2 strings can be concatenated eg 'xyz' + 'pqr' then string will be xyzpqr

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

class string1 {
private:
    string str;
public:
    void accept() {
        cout << "Enter a string: ";
        cin >> str;
    }
    void disp() {
        cout << "Concatinated string: " << str;
    }
void operator + (string1 & s1)
    {
        strcat(str, s1.str);
    }
};
int main()
```

```
{
    string s1, s2;
    s1.accept();
    s2.accept();
    s1 + s2;
    s1.display();

    return 0;

}
```

Output :-

Enter a string : xyz
Enter a string : pqr
Concatinated string : xyzpqr

2]

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

class I-login
{ protected
  string name,
  public:
  virtual void accept()
{ cout << "Enter Name:";
  cin >> name;
  cout << "Enter Password:";
  cin >> password;
}
   virtual void display
{ cout << "Name:" << name << end;
  cout << "Password : << password << endl;
};
};
class email_login : public I-login
  {   string email;
   public:
   void accept()

{ cout << "Email login details:" << endl;
  cout << "Name: << name << endl
  cout << "Password:" << password << endl;
  };
};
```

```
class membershiplogin : public Ilogn
{
    int member_id
    void accept ()
    {
    cout << "Membership login:"
     I_logn::accept ();
    cout << "Enter member ID:"
      cin >> member_id;
    }
    void display ()
{   cout << "Membership login Details:"
    cout << "Name:" << name << endl;
    cout << "Password:" << password << endl;
    cout << "Members ID:" member_id << endl;
    }
    };
int main()
{   I_login * login
    emaillogin e;
    membershiplogin m;

    login = &e;
    login -> accept ();
    login -> display ();
    login = &m;
    login -> accept ();
    login -> display ();

    return 0;

}
```

17/10

Experiment 9.

1.

```
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
   fstream new file;
new-file open ("new-file txt", ios :: out);
  if (! new file)
{
 count <<"file creation failed" << endl;
  }
   else
{
  cout << "New file created" << endl;
        new-file close ();
  }
   return 0;
  }
```

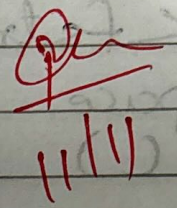
2.
```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
 ofstream file1("file1.txt", ios::app);
 if (!file1)
 {
  cout << "Error opening file1.txt"
       << endl;
  return 1;
 }
file1 << "Welcome to MIT" << endl;
file1.close();

 ifstream file1-read("file1.txt");
 ofstream file2("file2.txt");

 if (!file1-read)
  {
   cout << "Error opening files.txt
   for reading" << endl;
    return 1;
  }
   char ch;
  while (file1-read.get(ch))
 {
   file2.put(ch);
 }
```

```
cout << " content copied successfully
     from file1.txt to file2.txt"
      << endl;
file1.read.close();
return 0;
}
```

3.

```
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
ifstream inputfile ("input.txt");

if (! inputfile)
{
cout << "cannot open input.txt!" << endl;
return 1;
}

char ch;
int space count =0;
int digit count = 0;

while (input file.get (ch))
{
if (ch = = ' ')
space count ++;
else if (is digit (ch))
digit count ++;
}

inputfile.close ();
```

```
cout << "Total spaces : " << spacecount << endl;
cout << "Total digits : " << digitcount << endl;

return 0;
}
```



Experiment 10.

1.

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

template < class T>
T my function (T a [ ], int n) {
    T sum = 0;
   for (int i = 0; i<n; i++) {sum = sum
                              + a[i];
   }
    return sum;
}

int main () {
int a1[3] = {2, 0, 2};
float a2[3] = {1.2, 1.3, 2.4};
double a3[3] = {10.5, 20.5, 30.5};

count << "sum of integer array is:"
    << my function (a1, 3) << end l;
cout <<"Sum of float array is:"
    << my function (a2, 3) << end l;
cout << sum of double array is:"
    << my function (a3, 3) << end l;

  return 0;
}
```

2.

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

template <class T>
T square (T num) {
return num * num;
}

template < >
string square <string> (string str) {
    return str + str;
}

int main ( ) {
   int i = 5;
string s = "Hello";

cout << "Square of integer "<< i <<
<< square (i) << end l;

cout << " Square of string \""<< s <<
    << square(s) << end l;

return 0;
}
```

3.

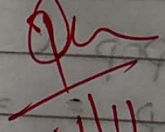
```
#include <iostream>
using namespace std;

template <class T>
class calculator {
    T a, b;
public:
    Calculator (T x, Ty) {
        a = x;
        b = y;
    }

    T add () {return a+b; }
    T sub () {return a-b; }
    T mul () {return a*b; }
    T div () {
        if (b != 0)
            return a/b;
        else {

        cout << "Error! Division by zero!"
             << endl;
        return 0;
    }
};
```

```
int main () {
    Calculator <int> c1 (10, 5);
cout << "Addition = " << c1. add() << endl;
cout << "Subtraction = " << c1. sub() << endl;
cout << "Multiplication = " << c1. mul() << endl;
cout << "Division = " << c1. div() << endl;

return 0;

}
```

4.

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

template <class T>
class stack {
  T stk [10];
  int top;
public;
  stack ( ) { top = -1; }

  void push (T val) {
    if (top = = 9)
    cout << "stack Overflow!" << end 1;
    else
      stk [++ top] = val;
  }
  void pop ( ) {
  if (top = = -1)
  cout << "Stack Underflow!" << end 1;
else
  cout << "Popped: " << stk [top - -] << end 1;
}

void display ( ) {
  if (top = = -1)
      cout << "stack is Empty!" << End 1;
else {
```

```
    cout << " stack: ";
for (int i = top ; i >= 0; i--)
        cout << stk[i] << " ";
cout << endl;
    }
  }

};

int main () {
 stack <int > s;
s. push (10);
s. push (20);
s. push (30);
s. display ();
 s. pop ();
 s. display ();

return 0;

}
```



Experiment #11

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

int main () {
vector <int> v = {1,2,3,4,5,6,7,8,9,10}

cout << "Initial Vector; " << end 1;
for (int i = 0 ; i < 10; i++ ) {
  cout << v [i] << "  " << end 1;
  }

cout << "Multiply by 10" << end 1;
for (int i = 0; i < 10 ; i++ ) {
  v[i] = v [i] * 10;
}

cout << "New Vector; " << end 1;
for (int i = 0; i <10; i++ ) {
 cout << v [i] << "  " << end 1;
}
return 0;

}
```

2. With using iterator

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

int main ()
{
  vector <int> v={ 1, 2, 3, 4, 5 }
cout << "vector before multiplication: ";
 for (vector <int> :: iterator it =
      v.begin , it = v.end (); ++it
 {
  cout << *it << " ";
 }
  cout << endl;
 int scalar;
 cout << "Enter scalar value: ";
  cin >> scalar;
  for (vector <int> :: iterator it = v.begin
    ; it != v.end (); ++it)
    {
      * it = (*it) * scalar;
    }
cout << "vector after multiplication: ";
for (vector <int> :: iterator it = v.begin()
   ; it != v.end (); ++it)
 {
    cout << *it << " "; }
   cout << endl;
   return 0;
}
```

Experiment - 12

a. WAP using STL to implement stack

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

int main ()
{
  stack <int> s;
  cout << "stack";
  s.push (10);
  s.push (20);
  s.push (30);

cout << "pop() << endl;
cout << "display elements";
 stack <int> = temp = s;
while (! temp.empty())
{
   cout << temp.top() << " ";
   temp.pop();
}
 cout <<endl;
cout << top element : << s.top();
 return 0;
}
```

b. WAP to imp STL to implement Que

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

int main(){
queue <int> q;
q.push(10)
q.push(20);
q.push(45);

cout << "Queue elements\n" <<endl;
queue <int> temp = q;
 while (!temp.empty())
 {
  cout << "item" <<temp.front() << " ";
     temp.pop();
 }
cout <<endl;
return 0;
}
```