PAGE No.
DATE / /

Exp 1

1) WAP to declare a class student having data members as name, Roll-no. Accept & display data for one student

```
#include <iostream>
#include <string>
using namespace std;
class student
{
    private:
    string name;
    int rollno;
    public:
    void userInputs()
    {
        cout << "Enter student details:" << endl;
        cout << "Enter student name:";
        cin >> name;
        cout << "Enter student rollnumber:";
        cin >> rollno;
    }
    void displayDetails()
    {
        cout << "Name:" << name << endl;
        cout << "Rollno:" << roll no << endl;
        cout << "\nStudent Details:" << endl;
    }
};
int main()
{
    student s1;
```

```
    s1.userInputs();
    s1.displayDetails();
    return 0;
}
```

Output

```
Enter student details:
Enter student name: heavenlight
Enter student rollnumber: 81
Name: heavenlight
Roll no: 81
```

3) WAP to declare data of H, M & S, Accept data for 1 object & display time in second.

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

class Time {
private:
    int h, m, s;
public:
    void inputTime(){
        cout << "Enter hours: ";
        cin >> h;
        cout << "Enter minutes: ";
        cin >> m;
        cout << "Enter seconds: ";
        cin >> s;
    }
    int toSeconds(){
        return h * 3600 + m * 60 + s;
    }
    void duplayTotalSecond(){
        cout << "Total time in seconds: " << toSecond() <<
        endl;
    }
};

int main(){
    Time t;
    t.inputTime();
    t.duplayTotalSecond();
```

return 0;
}

Output

Enter hour : 697 11
Enter minutes : 69 11
Enter seconds : 60 1
Total time in seconds : 40261

3) WAP to declare a class time having

2) WAP to declare a class Book having data members as id, name, price. Accept data for 2 books & display data of book having greater price.

```
#include<iostream>
#include<string>
using namespace std;
class Book
{
    public:
    int b_id;
    string b_name;

    int b_price;
    void taking_Inputs()
    {
    cout<<"Enter book details:"<<endl;
    cout<<"Enter book name:";
    cin >> b_name;
    cout<<"Enter book ID:";
    cin>>b_id;
    cout<<"Enter book ID:";
    cin>>b_id;
    cout<<"Enter book price:";
    cin>>b_price;
    }
};
int main()
{
    Book obj1, obj2;
```

```
obj1.taking_Inputs();
obj2.taking_Inputs();
if(obj1.b_price > obj2.b_price)
{
    cout << "Name: " << obj1.b_name << endl;
    cout << "ID: " << obj1.b_id << endl;
    cout << "price: " << obj1.b_price << endl;
}
else
{
    cout << "Name: " << obj2.b_name << endl;
    cout << "ID: " << obj2.b_id << endl;
    cout << "Price: " << obj2.b_price << endl;
}
return 0;
}
```

Output

```
Enter book details:
Enter book name: hzhgj
Enter book ID: 794
Enter book price: 974898
Enter book details:
Enter book name: ghuxxxu
Enter book ID: 787
Enter book price: 787
Name: hzhgj
ID: 794
Price: 974898
```

25/7/25

Exp 2

PAGE No.
DATE / /

2) Write a c++ program to demostrate the use of array of objects.

a) WAP to declare a class 'city' having data member as name and population. Accept the data for 5 Cities and display name of city having highest population.

```
#include <iostream>
using namespace std;
class city
{
  public:
  string name;
  int population;
  void accept ()
  {
   cout << "Enter city Name: ";
   cin >> name;
   cout << "Enter city Population: ";
   cin >> population;
  }
  void disp ()
  {cout << "city having highest population : " << name; }
int main()
{ city c[5];
int i, max;
for (i = 0; i < 5; i++)
{ c[i] . accept (); }
max = c[0] . population;
for (i = 0; i < 5; i++)
{ if (c[i] . population > max)
```

```
{max=i;}}
c[max].disp();
return 0;
}
```

OUTPUT

```
Enter city Name: Mwanza
Enter city Population: 789
Enter city Name: Pune
Enter City Population: 567
Enter City Name: Mbeya
Enter City Population: 389
Enter City Name Kharagpur
Enter City Population: 456
Enter City Name: Kolkata
Enter City Population: 989
City having highest population: Kolkata
```

b) WAP to declare a class 'Account' having data members as Account no. and balance. Accept this data for 10 accounts and give interest of 10% where balance is equal or greater than 5000 and display them

```
#include<iostream>
using namespace std;
class Account{
public:
  int account_no;
  float balance;
```

```
void accept(){
    cout<<"Enter account number and balance:";
    cin>>account_no>>balance;
}
void applyInterest(){
    if(balance>=5000){
        balance+=(balance*0.10);
    }
}
void display(){
    cout<<"Account Number:"<<account_no<<"Balance
    after Interest:"<<balance<<endl;
}

};
int main(){
  Account accounts[10];

for(int i=0;i<10;i++){
    accounts[i].accept();
}

cout<<"\nAccounts with balance>=5000 after applying
        interest:\n";
for(int i=0;i<10;i++){
    accounts[i].applyInterest();
    if(accounts[i].balance>=5000){
        account[0].display();
    }
}
return 0;
}
```

OUTPUT

Enter account number and balance: 1
599
Enter account number and balance: 2
799
Enter account number and balance: 3
999
Enter account number and balance: 4
4999
Enter account number and balance: 5
399
Enter account number and balance: 6
799
Enter account number and balance: 7
888
" " " " "
Enter account number and balance: 10
Account with balance >= 5000 after applying interest

c) WAP to declare a class 'staff' having data members as name and post. Accept this data for 5 staff and display names of staff who are "HOD"

```
#include<iostream>
using namespace std;
class staff
{ string name;
public:
string post;
void accept()
{
cout<<" Enter the staff name:";
getline(cin, name);
```

```
cout<<"Enter the staff post name:";
getline(cin, post);
}

void disp()
{cout<<"\n Staff Name:"<<name;
 cout<<"\n Staff Post:"<<post;}};
int main()
{
 staff s[5];
 int i;
 for(i=0; i<5; i++)
 { s[i].accept(); }
 for(i=0; i<5; i++)
 {if(s[i].post=="HOD")
 {s[i].disp();}}
 return 0; }
```

Output

```
Enter the staff name: Jim Daha
Enter staff post: lecturer
Enter the staff name: Alex Dontoliver
Enter the staff post: HOD
Enter the staff name: Travis Scott
Enter the post: Dean
Enter the staff name: Herbert Jochovik
Enter the staff post: HOD
Enter staff name: Donald Jordan
Enter the staff post: Principal

Staff Name: Alex Dontoliver
Staff Post: HOD
Staff Name: Herbert Jochovik
Staff post: HOD
```

25/7/25

Exp: 3

1) a) Write a program to declare a class 'book' containing data member as book-title, author-name and price. Accept and display the information for one object using pointer to object.

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

class Book {
    string book_title;
    string author_name;
    float price;
public:
    void accept() {
        cout << "Enter book title: ";
        getline(cin, book_title);
        cout << "Enter author name: ";
        getline(cin, author_name);
        cout << "Enter price: ";
        cin >> price;
        cin.ignore();
    }
    void display() {
        cout << "\nBook Title: " << this->book_title << endl;
        cout << "Author Name: " << this->author_name << endl;
        cout << "Price: $" << this->price << endl;
    }
};
int main() {
    Book b;
    Book *ptr = &b;
    ptr->accept();
    ptr->display();
    return 0;
}
```

Output

Enter book title: Zach
Enter author name: z
Enter price: 300

Book Title: Zach
Author Name: z
Price: $300.

b) WAP to find the area declare a class 'student' having a data member roll_no and percentage. Using 'this' pointer invoke member function to accept and display the data for one object of the class

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

Class student {
    int roll_no;
    float percentage;

public:
  void accept (int roll_no, float percentage){
        this->roll_no = roll_no;
        this->percentage = percentage;
  }

  void display () {
      cout<<" Roll No: <<this->roll_no<<endl;
      cout<<"Percentage: "<<this->percentage<<endl;
  }
};
```

```
int main() {
    student s1;

    int r;
    float p;

    cout << "Enter roll number :";
    cin >> r;
    Cout << "Enter roll number :";
    cin >> r;
    cout << "Enter percentage :";
    cin >> p;

    s1.accept(r, p);
    Cout << "\nStudent Details:\n";
    s1.display();

    return 0;
    }
```

Output

```
Enter roll number :80
Enter percentage:90

Student Details:
Roll No:80
Percentage:90
```

Experiment no. 4

PAGE No. / DATE / /

Q1) Swap 2 no. from same class using object as function write Swap function as member function?

```
#include <iostream>
Using namespace std;
Class num {
Public:
 int n1, n2;
 void accept() {
    cout << " Enter first number: " << endl;
    cin >> n1;
    cout << " Enter Second number: " << endl;
    cin >> n2;
 }
 void display() {
    cout << " Num 1: " << n1 << endl;
    cout << " Num 2: " << n2 << endl;
 }
 void swap (num) {
    int temp = n1; n1 = n2; n2 = temp;
    cout << " Swapped: \n first number: " << n1 << "\n
        Second number: " << endl;
 }
};
int main() {
    no. accept();
    no. display();
    no. Swap (m);
    return 0;
}
```

(6)

Q2 Swap two no. from same class using concept of friend function

```
#include <iostream>
using namespace std;
class name {
public: int n1, n2;
        void accept(){
            cout<<" Enter first no :"<<endl;
            cin>>n1;
            cout<<" Enter second no :"<<endl;
            cin>>n2;
        }
        void display(){
            cout<<" Num 1 :"<<n1<<endl<<"Num 2:"<<
                n2<<endl;
        }
};no;
void swap(num){
    int temp=no.n1;
    no.n1=no.n2;
    no.n2=temp;
    cout<<" Swapped :\nFirst:"<<no.n1<<"\nSecond:
        "<<no.n2<<endl;
}
```

```
int main(){
    no.accept();
    no.display();
    Swap(no);
    return 0;
}
```

Output

Before Swapping:

num 1 = 10, num2 = 20

After Swapping:

num 1 = 20, num2 = 10

Q3) Swap two number from different class using friend function

```
#include<iostream>
Using namespace std;
class num{
public: int n1;
    void accept(){
        cout<<"Enter first no.:"<<endl;
        cin>>n1;}
    void display(){cout<<"Num 1:"<<n1<<endl;}
};no1;
class num2{
public: int n2;
        void accept(){
            cout<<"Enter Second no.:"<<endl;
            cin>>n2;}
        void display(){cout<<"Num 2:"<<n2<<endl;}
void Swap(num&1 num2){
```

```
int temp = no1. n1;
no1.n1 = no2. n2;
no2 n2 = temp. ;
cout << "Swapped:\n first:" << no1.n1 << "\n
    second: " << no2.n2 << endl;
}
int main (){
    no1.accept ();
    no2.accept ();
    no1. display ();
    no2. display ();
    swap (no1, no2);
    return 0;
}
```

Output
Enter first no: 5
Enter second no.: 10
Num 1: 5
Num 2: 10
Swapped
In first: 10
In second: 5

4) Create 2 classes Result1 & Result2 which states the marks of the students. Read the value of marks for both the class objects & concept avg.

```
#include<iostream>
using namespace std;
class Result{
public;
int r1;
  void accept () {
      cout<<" Enter first result :"<<endl;
      cin>>r1; }
  }res1;
  class Result 2{
  public:
     int r2;
     void accept () {
        cout<<"Enter Second result:"<<endl;
          cin>>n2; }
     }res2;
  float avg ( Result1, Result2){
     return (res1.r1 + res2.r2)/2;
  }
  int main () {
       res1. accept ();
       res2. accept ()
       float avg= avg (res1, res2);
       cout<<"Average:"<< average;
       return 0;
  }
```

Output

Enter first result:
Enter second result.
Average : 85

Q5) Find the greatest number among 2 numbers from 2 different classes using friend fun?

```
#include <iostream>
using namespace std;
class num1 {
public: int n1;
        void accept() {
            cout << "Enter first no. :" << endl;
            cin >> n1; }
} no1;
class num2 {
public: int n2;
        void accept() {
            cout << "Enter second no. :" << endl;
            cin >> n2; }
} no2;
int gnum(int n1, int n2) { return (n1 > n2 ? n1 : n2); }
int main() {
    no1.accept();
    no2.accept();
    int gr = gnum(no1.n1, no2.n2);
    cout << "Greatest: " << gr << endl;
    return 0;
}
```

EXPERIMENT 5

Write a C++ program to implement types of constructors

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

```
#include <iostream>
using namespace std;
class sum{
    int n, total;
public:
    sum(int num){
        n = num;
        total = 0;
        for (int i = 1; i <= n; i++){
            total += i;
        }
    }
    void display(){
        cout << "Sum of numbers from 1 to " << n << ": "
             << total << endl;
    }
};
int main(){
    int n;
    cout << "Enter number: ";
    cin >> n;

    sum s(n);
    s.display();
    return 0;
}
```

OUTPUT

Enter number: 5

Sum of number from 1 to 5 is : 15

5) WAP to declare a class 'student' having data members as name and percentage. Write a constructor to initialize these data members. Accept & display data for one student

```
# include <iostream>
# include <string>
using namespace std;
class student
{
  private:
  string name;
  float percentage;
  Public:
  student (string n, float p)
  {
    name = n;
    percentage = p;
    cout << "Name:".
    getline (cin, name);
    cout << "Percentage:";
    cin >> percentage;
  }
```

```
void display()
{
  cout << "student name:" << name << endl;
  cout << "student percentage" << percentage << endl;
  }
};
int main()
{
  student S1 ("Heaven", 89.4);
  S1.display();
  return 0;
}
```

Output

Name : Heaven
Percentage : 89.4
student Name : Heven
student Percentage 89.4

c) Define a class 'college' members variables as roll_no, name course. WAP using constructor with default value as "Computer Engineering" for course. Accept this data for two objects of class and display the data

```
#include <iostream>
#include <string>
using namespace std;
class college
{
    private:
    string course_name, stud_name;
    int roll_no;
    public:
    College()
    {
      roll_no=80;
      stud_name="Heavenlight";
      course_name="CSF";
    }
    void display()
    {
      cout<<"Roll no:"<<roll_no<<endl;
      cout<<"student name:"<<stud_name<<endl;
      cout<<"course name:"<<course_name<<endl;
    }
};
int main()
{
  college c1;
  c1.display();
  return 0;
}
```

OUTPUT
Roll no: 80
Student name: Heavenlight
Course name: CSE

4) Write a program to demostrate constructor overload

```
#include <iostream>
using namespace std;
class rectangle
{
  int l, b;
  public:
  rectangle ()
  {
      l=2;
      b=5;
  }
  rectange (int x)
  {
      l=x;
      b=x;
  }
  rectangle (int x, int y)
  {
      l=x;
      b=y;
  }
  rectangle (rectangle &r3)
  {
    l=r3.l;
    b=r3.b;
```

```
}
void calculate()
{
    cout<<"The area is :"<<(l*b)<<endl;
}

};
int main()
{
  rectangle r1;
  r1.calculate();
  rectangle r2(3);
  r2.Calculate();
  rectangle r3 (6,6);
  r3.calculate();
  rectangle r4(r3);
  r4.Calculate();
  return 0;
```

OUTPUT

The area is : 10
The area is : 9
The area is : 36
The area is 36

17/10

EXPERIMENT 6

1a) WAP to implement multi level inheritance, Assume suitable data

```
#include <iostream>
#include <string>
using namespace std;
class Person
{
  protected:
   int age;
   string name;
};
class student: Public Person
{
  private:
  int roll_no;
  public:
  void accept()
  {
   cout << " Enter your name:";
   cin >> name;
   cout << " Enter your age:";
   cin >> age;
   cout << "Enter your roll no:";
   cin >> roll_no;
  }
  void display()
  {
   cout << " Name \n" << name;
   cout << " Age \n" << age;
    cout << "Roll no\n" << roll_no;
  }
};
```

```
int main()
{
 student s1.
 s1.accept();
 s1.display();
 return 0;
}
```

b) Write a program to implement multiple. Assume suitable data

```
#include<iostream>
using namespace std;
class Academic
{
 protected;
 int marks;
};
class Sports
{
  protected:
   int score;
};
class Result : Protected Academic sports
{
  int total_score=0;
  public:
  void accept()
  {
    cout<<"Enter the marks of the student:";
    cin>>marks;
```

```
cout << "Enter the sports score of the student";
cin >> score;
}
void calculate()
{
total_score = marks + score;
cout << "The marks of the student is :" << marks << endl;
cout << "The sport score of the student is: " << score << endl;
cout << "The total score of the student is: " << total_score << endl;
}
};
int main()
{
Result r;
r.accept();
r.calculate();
return 0;
}
```

OUTPUT

```
Enter the marks of the student: 99
Enter the sports score of the student: 95
The marks of students i : 99
The sport score of the student i : 95
The total score of the student i : 194
```

PAGE NO.
DATE / /

C. WAP to implement hierarchial inheritance. Assume suitable data

```
#include<iostream>
using namespace std;
class vehical
{
  protected:
  string brand;
  int model;
};
class car: protected vehical
{
  protected:
  string type;
};
class Electric Car: Protected
{
  int batterycapacity;
  public:
  void accept()
  {
    cout<<"Enter the brand of car:";
    cin>>brand;
    cout<<"Enter the model of the car:";
    cin>>model;
    cout<<"Enter the type of car:";
    cin>>type;
    cout<<"Enter the battery capacity of the car:";
    cin>>battery capacity;
  }
```

```
void display()
{
    cout<<"\nThe brand of the car:"<<brand<<endl;
    cout<<"The model of the car:"<<model<<endl;
    cout<<"The type of the car:"<<type<<endl;
    cout<<"The battery capacity of the car:"<<batteryCapacity<<endl;
  }
};
int main()
{
  Electriccar e;
  e.accept();
  e.display();
  return 0;
}
```

Output:

```
Enter the brand of the car: BMW
Enter the model of the car: M4
Enter the type of the car: Sedan Sport
Enter the battery capacity: 115
```

PAGE No.
DATE / /

4. WAP to implement hybrid inheritance

```
#include<string>
#include<iostream>
using namespace std;
class Person
{
public:
string name;
int age;

void getPersonDetails()
{
    cout<<"Enter name:";
    cin>>name;
    cout<<"Enter age:";
    cin>>age;
}
void showDetails()
{
 cout<<"Name"<<name<<", Age:"<<age<<endl;
}
};
class student: Public Person
{
public:
string course;
 void getDetails()
 {
    cout<<"Enter course:";
    cin>>course;
 }
```

```
void showDetails()
{
  cout << " course: " << course << endl;
}
};
class Employee: Public Person
{
Public:
string company;
void getEmployeeDetails()
{
   cout << " Enter Company: ";
   cin >> company;
}
void showEmployeeDetails()
{
  cout << " Company: " << company << endl;
}
};
class Intern: Public student, public Employee {
public:
   void showInternDetails()
   {
     cout << "\n --- InternDetails --- \n";
     student::showPersonDetails();
     showEmployeeDetails();
   }
};
int main()
{
  Intern it;
  it.student!! getPersonDetails();
```

it.getDetails();
it.getEmployeeDetails();
it.showInternDetails();

return 0;
}

O/P
Enter name: Heavenlight
Enter age: 45
Enter course: Computer Science
Enter company: Microsoft

- - - Intern Details - - -
Name: Heavenlight, Age: 45
Course: Computer Science
Company: Microsoft

Q
17/10

Experiment 7

a) WAP using function Overloading to calculate area of laboratory (rectangular in shape) & area of classroom

```
#include<iostream>
using namespace std;
Class Area;
{
 private:
 float l, b;
 public:
 void area (float l)
 {
    float area;
    area = l*l.
    cout << "Area of square: " << area << endl;
 }
 void area (float l, float b)
 {
    float area;
    area = l*b;
    cout << "Area of rectange: " << area;
 }
};
 int main()
 {
    Area a1;
    a1.area(8);
    a1.area(4,12);
    return 0;
 }
```

OXP

Area of square : 64
Area of rectangle: 48

PAGE No.
DATE / /

5) WAP using function overloading to Calculate the sum of 5 float value & 10 int

```
#include <iostream>
using namespace std;
Class Sum
{
private:
 int a,b,c,d,e,f,g,h,i,j;
 float k,l,m,n,o;
public:
Void sum (float k, float l, float m, float n, float o){
 float sum;
 sum= k+ l+ m +n+ o;
 cout<<" Sum of 5 floating numbers :"<<sum<<endl; }
 void sum(int a, int b, int c, int d, int e)
int f, int g, int h, int i, int j ) {
 int sum;
 sum = a + b + c + d + e+ f+ g+ h+ i+ j;
 cout<<"sum of 10 integers :"<<sum;
 }
};
int main()
{
 Sum s1;
 s1.sum( 1.2, 3.5, 5.6, 8.4, 1.5);
 s1.sum( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
 return 0;
}
```

Output:
Sum of 5 floating numbers: 20.2
sum of 10 integers: 55

c) WAP to implement unary - operator when used with the obj so that the numeric data member of the class is negated

```
#include<iostream
using namespace std;
class Number
{
   private: int x;
   public: void accept()
   {
      cout << "Enter a number:";
      cin >> x;
   void operator -()
   {
      x = -x;
   }
   void display()
   {
   int main()
   {
      Number n1;
      n1.accept();
      -n1;
      n1.display();
      return 0;
   }
```

Output
Enter a number: 5
Negated number: -5

PAGE NO.
DATE / /

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

```
#include<iostream>
using namespace std;
class Number
{
 private: int x; int accept;
 public: void accept()
 {
   cout<<"Enter a number";
   cin>>x;
 }
temp=x;
 void operator ++()
{
   x=++x;
}
void reset()
{
   x = temp;
}
void operator ++(int)
{
   x=x++;
}
void display()
{
 cout<<"(pre) The number is :"<<x<<endl;
}
```

```
void display2()
{
  cout << "(post) The number is : " << x;
  }
};
int main()
{
  Number n1;
  n1.accept();
  ++n1;
  n1.display();
  n1.reset();
  n1++;
  n1.display2();
  return 0;
}
```

Output

Enter a number : 5

(pre) The number is: 6

(post) The number is: 5

Experiment- 8

a) WAP to overload '+' operator so that two strings can be concatenated. Eg: "xyz" + "pqR" = "xyzpqR"

```
#include <iostream>
#include <string>
using namespace std;
class Mystring {
public: string str;
 My string operator + (Mystring &other) {
   Mystring temp;
    tempstr = str + otherstr;
    return temp;
  }
  void display()
  {
   cout << str << endl;
  }
};
int main()
Mystring s1, s2, s3;
s1.str = "xyz";
 s2.str = "pqr";
s3 = s1 + s2;
cout << " concatenated string:";
s3.display();
return 0;
}
```

output

Concatenated string:

b) Write a program 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 membership login details of employee.

```
#include <string>
#include<iostream>
using namespace std;
class login{
   protected: string name, password;
   public: virtual void accept(){
            cout << "Enter name:";
            cin >> name;
            cout << "Password:";
            cin>>password;
          }
};
class Email login: Public login{
      string email;
    public:
    void accept() override{
    Login::accept()
    cout <<" Enter email:";
    cin >> email;
    }
    void display(){
      cout <<"\n ---- Email login Details ---\n";
      cout << "Name: " << name << " Password :" <<
      password << Email: " << email << endl;
    }
};
```

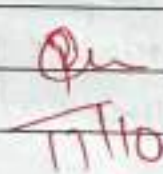
```
class MembershipLogin : public Login{
    string membership ID;
    public:
    void accept() Override{
    Login : accept();
    cout << " Enter membership ID: ";
    cin >> membership ID;
    }
    void display(){
    cout << "\n --- Membership Login Details ---\n";
    cout << "Name: " name << \npassword: " << password <<
     "Membership ID:" << membership ID << endl;
    }
};
int main()
{
    Email Login e;
    Membership login m;
    e.accept();
    m.accept();
    e.display();
    m.display();
    return 0;
}
```

Output

Enter Name : Heavenlight
Enter Password 125
Enter email : abc@gmail.com
Enter name : Heavenlight
Password : 754
Enter membership ID: 7545

PAGE No.
DATE / /

--- Email login Details ---
Name : Heavenlight
Password : 125
Email : abd@gmail.com
--- Membership Login Details ---
Name : Heavenlight
Password: 547
Membership ID: 7545



Experiment - 9

a) WAP to copy the contents of one file into another

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  fstream first_file;
  fstream second_file;
  first_file.open("First.txt", ios::in);
  if(!first_file)
  {
    cout<<"Error opening second.txt!"<<endl;
    return 1;
  }
  char ch;
  while(first_file.get(ch))
  {
    second_file.put(ch);
  }
  first_file.close();
  second_file.close();
  cout<<"file copied successfully!"<<endl;
  return 0;
}
```

5) WAP to count digits & spaces using file handling

```
#include<iostream>
#include<fstream>
#include<cctype>
using namespace std;
int main()
{
  fstream new_file;
  new_file.open("Fist.txt", ios::in);
  if (!new_file)
  {
    cout << "Error Occured while opening!" << endl;
    return 1;
  }
  int digits_count=0;
  int spaces_count=0;
  char ch;
  while (newfile.get(ch)){
    if(isdigit(ch))
    {
      digits.count++;
    }
    if(isspace(ch)){
      spaces_count++;
    }
  new_file.close();
  cout << "Number of digits: " << digits_count << endl;
  cout << "Number of spaces: " << spaces_count << endl;
  return 0;
}
```

PAGE No.
DATE / /

Q) WAP to count words using file handling

```
#include<iostream>
# include< fstream>
# include <cctype>
using namespace std;
{
 fstream new_file;
 new_file.open("Fruit.txt", ios::in);
 (!new_file)
 {
 cout<<" Error Occured!"<<endl;
 return 1;
 }
 char ch;
 int words_count=0;
 bool inWord=false;
 while (new_file.get(ch))
 {
  if (isspace(ch))
  {
    inWord = false; }
  else if(!inWord)
  {
    words_count++;
    inWord=true;
  }
 }
 new_file.close()
 cout<<" Number of words: "<<words_count<<endl;
 return 0;
}
```

Q.) WAP to count occurance of given word

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
int main()
{
    fstream new_file;
    new_file.open("Fruit.txt", ios::in);
    if(!new_file)
    {
     cout<<"Error in opening Fruit.txt!"<<endl;
     return 1;
    }
    string target = "Heaven";
    string word;
    int count=0;
    while(new_file>>word)
    {
      if(word==target)
      {
        count++;
      }
    }
    new_file.close();
    cout<<"The word: "target<<"\" Occured "<<
      count<<" times"<<endl;
    return 0;
}
```

Experiment-10

Q: WAP to find sum of array element using function template (eg: for integer, float & double array of 10 elements)

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

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

int main()
{
    int size = 10;
    int intArr[size] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    float floatArr[size] = { 1.1, 2.2, 3.2, 4.4, 5.5, 6.6, 7.7,
                             8.8, 9.9, 10.10};
    double doubleArr[size] = { 1.11, 2.22, 3.33, 4.44, 5.55,
                               6.66, 7.77, 8.88, 9.99, 10.10};

    cout << "sum of integer array = " << findSum(intArr, size)
         << endl;
    cout << "sum of float array = " << findsum(floatArr, size);
    Cout << " sum of doubleArr = " << findsum(doubleArr, size)
         << endl;
    return 0;
}
```

O/P:-
Sum of integer array = 55
Sum of float array = 59.6
Sum of double array = 60.05

Q2. WAP to square function using template specialization

```
#include <bits/stdc++.h>
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 intNum = 5;
  string str="Hello";

cout << "Square of integer" << intNum << "=" << square
(intNum) << endl;
cout << "square of string\" << str << "\"=" << square(str)
<< endl;
return 0;
}
```

O/P
Square of integer = 25
Square of string= HelloHello

3. WAP to build a simple calculator using class template

```
#include<bits/stdc++.h>
using namespace std;

template<class T>
class Calculator{
  private:
    T num1, T num2;
  public:
    calculator (T n1, T n2){
    num1 = n1;
    num2 = n2;
    }
T add(){ return num1 + num2; }
T subtract() { return num1 - num2; }
T multiply(){ return num1 * num2; }
T divide(){
    if (num2 != 0)
      return num1 / num2;
    else {
     cout << " Error! division by zero" << endl;
        return 0;
        }
    }
  Void displayResult(){
    cout << "Numbers: " << num1 << " and " << num2 << endl;
    cout << " Addition = " << add() << endl;
    Cout << "Substraction" << subtract() << endl;
    cout << " Multiplication = " << multiply() << endl;
    Cout << "Division = " << divide() << endl;
    }
};
```

```
int main(){
  cout << "Simple Calculator using class Template \n";
  calculator<int> intCalc(10,5);
  cout << "\n Integer calculator --- \n";
  int Calc.displayResults();
  calculator<float> floatCalc(10.5, 2.5);
  cout << "\n --- float calculator --- \n";
  float calc.displayResults();
  return 0;
}
```

O/P

Simple calculator using class Template

--- Integer Calculator - - -

Number : 10 and 5

Addition = 15

Subtraction = 5

Multiplication = 50

Division = 2

--- float Calculator - - -

Number : 10.5 and 25

Addition = 13

Subtraction = 8

Multiplication = 26.25

Division = 4.2

4. WAP to implement push & pop methods from stack using class template

```
#include <bits/stdc++.h>
using namespace std;
#define SIZE 5
template<class T>
class stack{
  private:
  T arr[size];
     int top;
  public:
   stack(){
    top=-1
        }
 void push(T value){
  if (top==size-1){
    cout<<"stack overflow! cannot push"<<
        value<<endl;
  }
  else{
    arr[++top]=value;
    cout<<value<<"Pushed into stack"<<endl;}}
void pop(){
   if(top==-1){
     cout<<"stack is empty "<<endl;}
   else{
    cout<<arr[top--]<<popped from stack"<<endl;
     }
   }
   void display(){
      if(top==-1){
       cout<<"stack is empty"<<endl;}
     else{
      cout<<"stack is empty "<<endl;}
```

```
else {
  cout<<"stack elements:";
  for (int i= top; i>=0; i--){
      cout<<arr[i]<<" ";}
      cout<<endl;
      }
  }
};
int main(){
  cout<<" stack implementation using class template
        \n";
stack<int>int stack;
cout<<"\n --- Integer stack Operation --\n";
int stack.push(10);
int stack.push(20);
int stack.push(30);
int stack.display();
int stack.pop();
int stack.display();

stack<string>strstack;
cout<<"\n --- string stack operation --\n";
str stack.push("Apple");
str.stack.push("Banana");
strstack.display();
str.stack.pop();
str.stack.display();
  return 0;
}
```

PAGE No.
DATE / /

Q/P

--- Integer stack Operations ---

10 pushed into stack
20 pushed into stack
30 pushed into stack
stack elements: 30 20 10

PAGE No.
DATE / /

Exp: 11

Accessing vector using iterator

```
# include <bits/stdc++.h>
using namespace std;
int main()
{
  vector <char> v(10);
  vector <char>::iterator p;
  int i;
  p = v.begin();
  i = 0;
  while (p != v.end()) {
    *p = i + a;
    p++;
    i++;
  }
  cout << "Original Content:\n";
  p = v.begin();
  while (p != v.end()) {
    cout << *p << " ";
    p++;
  }
```

o/p

Vector contents

a b c d e f g h i j

Without iterator

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
vector<char> v(10);
unsigned int i;
cout<<"size="<<v.size()<<endl;
for(i=0;i<10;i++)
{
  v[i]=i+a;
}
cout<<"Current content:"
for(i=0;i<v.size();i++){
cout<<v[i]<<" ";}

cout<<"\n\n".
cout<<"Expanding vector:\n";
for(i=0;i<10;i++)
{
  v.push_back(i+10+'a');}

cout<<"size now="<<v.size()<<endl;
cout<<"current contents:\n".
for(i=0;i<v.size();i++)
{
  v[i]=toupper(v[i]);
}
cout<<"Modified Contents:\n".
for(i=0;i<v.size();i++)
{
  cout<<v[i]<<" ".
}
```

}
cout<<endl;
return 0;
}

o/p

Current elements in vector are:
a b c d e f g h i j

Expanding Vector

Modified contents:
A B C D E F G H I J

Exp - 12

a) Implement stack

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
  stack <char> cars;
  cars.push("BMW");
  car.push("Audi");
  car.push("Mercedes");
  cars.push("Ferrari");
  cout<<"Top element : "<<cars.top()<<endl;
  cout<<"size of stack is : "<<cars.size()<<endl;

  car.pop();
  car.pop();
  while(!cars.empty())
  {
    cout<<"Elements in stack are: "<< cars.top()
    <<endl;
    car.pop();
  }
  return 0;
}
```

O/P

Top element : Ferrari
size of stack: 4
Elements in stack are : Audi
BMW

5) Implement Queue

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  queue<int>age;
  age.push(21);
  age.push(22);
  age.push(23);
  age.push(24);
  cout<<"Front element is :"<<age.front()<<endl;
  cout<<"Back element is :"<<age.back()<<endl;

  age.pop();
  age.pop();
  while(!age.empty())
  {
    cout<<"Element in queue are:"<<age.front()<<endl;
  age.pop();}
  return 0;
}
```

O/P

Front element : 21
Back element : 24