* Write C++ Codes *

Page No.:
Date:

1] To add 2 numbers

→
```
#include <iostream.h>
int main()
{
int a, b, sum;
std:: cout<< "Enter any 2 numbers \n";
std:: cin>> a >>b;
sum = a+b;
std:: cout << "Sum is \t" << sum;
return 0;
}
```

* Output

```
Enter any 2 numbers
5
6
Sum is 11
```

2] Arithmetic operations using switch.

→
```
#include<iostream>
int main()
{
float a, b;
char ope;
std:: cout << " +, -, /, * ";
std :: cin >> a>> b;
switch (ope)
{
case '+' : std:: cout << "Sum is " << a+b : break;
case '-' : std :: cout<<"Substraction is" << a-b : break;
case '*' : std :: cout << " Multiplication is " <<a*b : break;
case '/' : if (b!=0)
```

```
std::cout << "Division is <<a/b;
else
std::cout<< "Can't divide by zero";
break; }
return 0;
}
```

* Output

```
+, -, *, / : /
Enter two number : 5
4
Division is 1.25
```

3] Check no. is even or odd.

→
```
#include <iostream>
 int main()
{
int num, even, odd;
std::cout << "Enter an integer:";
std:: cin >> num;
if (num % 2 == 0)
{
std::cout<<num<<" is even".;
} else
else { std: cout <<num << is odd."; }
return 0; }
```

* Output

```
Enter an integer 5
5 is odd
```

4] Print 1 to 10 no. using

i) For loop

→
```
#include <iostream>
int main()
{
int i;
for (i=1; i<=10; i++)
{
std::cout<<"\n"<<i;
}
return 0;
}
```

ii) While loop

→
```
#include <iostream>
int main()
{
int i=1;
while (i<=10)
{
std::cout<<"\n"<<i;
i++;
}
return 0;
}
```

* Output
1
2
3
4
5
6
7
8
9 10

5] a)
```
  *
 * *
* * *
```

→
```
#include <iostream>
int main()
{
int i,j,s;
for(i=1; i<3; i++)
{
for(s=1; s<=3-i; s++)
{
std::cout<<" ";
std::cout
}
for(j=1; j<=i; j++)
{
std::cout<<" * ";
std::cout<<" ";
}
std::cout<<"\n";
}
return 0;
}
```

b)
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

→
```
#include <iostream>
int main()
{
int i,j;
for (i=1 ; i<=5 ; i++)
 {
for (j=1 ; j<=i ; j++)
{
std:: cout << j ;
 }
std :: cout <<"\n";
 }
return 0;
}
```

* Output

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

26/8

Experiment - 1

1] Write a program to declare class students having data members as roll no., name accept & display data for 1 object.

→

```
#include <iostream>
using namespace std;
class student
{
int roll;
string name;
public:
void accept()
{
cout<<"roll:"<<roll<<", Name:"<<name
<<endl;
}
};
int main()
{
student s1;
s1.accept();
s1.display();
return 0;
}
```

★ Output

```
Enter value of roll, name 42
Siddhant
roll: 42, name: Siddhant
```

2] Write a program to declare a class book having data members as book name, prise, no. of pages, accept for 2 obj. & display the name of book having greater prise.

→

```
#include <iostream>
using namespace std;
class book
{
int pages;
string name;
float price;
 public:
 void accept()
 {
cout<<" Enter value of name, prise, pages";
cin >> name >> prise >> pages;
 }
 void display ()
 {
cout<<" Name: "<< name <<", Prise: "<<prise
    <<" Pages:" << pages << endl;
 }
};
int main()
{
book b1, b2;
b1. accept();
b2. accept()
cout<<"\n Book with greater prise:\n";
```

2] Write a program to declare a class book having data members as book name, prise, no. of pages, accept for 2 obj. & display the name of book having greater prise.

→

```
#include<iostream>
using namespace std;
class book
{
int pages;
string name;
float price;
 public:
 void accept()
 {
cout<<" Enter value of name, prise, pages ";
cin >> name >> prise >> pages;
 }
 void display ()
 {
cout<<" Name: "<<name<<", Prise: "<<prise
    <<" Pages:" << pages << endl;
 }
};
int main()
{
book b1, b2;
b1. accept();
b2. accept()
cout<<"\n Book with greater prise:\n";
```

```
if (b1.prise > b2.prise)
{
b1.display();
}
else if (b2.price > b1.price)
{
b2.display();
}
else
{
cout<<"Both books have the same price:\n";
    b1.display();
    b2.display();
    }
    return 0;
    }
```

* Output

```
Enter value of name, price, pages: Harry Potter
        500
        300
Enter value of name, price, pages: Famous Five
        650
        340
Book with greater price:
Name: Famous Five , price: 650 , pages: 340
```

3] Write a program to declare a class time, accept the time in HH:MM:SS & convert it into seconds & display them.

—>

```
#include <iostream>
using namespace std;
class Time
{
private:
int H, M, S ;
public:
void accept ()
{
cout <<" Enter Time";
cin >> H>> M >> S;
}
void display ()
{
int total;
total = ((H * 3600) + (M * 60) + S);
cout <<" Total time in seconds " << total;
}
};
int main()
{
Time t1 ;
t1. accept();
t2. display ();
return 0;
}
```

Q1 — 26

* Output

Enter time 12
56
45 Total time in seconds 46605

Experiment - 2

1] Write a program to declare a class 'city' having data members as name & population. Accept this data for 5 cities & diplay name of city having highest population.

—>

```
#include <iostream>
using namespace std;
class city
{
int Population;
String name;
public :
void accept ()
{
cout<<"Enter city name & population:";
cin>>> name>> population;
 }
};
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++)
{
int (c[i] . population > max)
{
```

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

* Output

```
Enter city name & population: pune
25000
Enter city name & population: mumbai
20000
Enter city name & population: satara
18000
Enter city name & population: banglore
30000
Enter city name & population: nagpur
10000

City with highest population:
name: Banglore , population: 30000
```

2] Write a program to declare a class "Account" having data members as account no. & balance. Accept this data for 10 accounts & give interest of 10% where balance is equal or greater than 5000 & display them.

→

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

  void accept()
  {
 cout<<" Enter name of customer:";
 cin >> name;
 cout <<" Enter account_number:";
 cin >> acc_no;
 cout <<" Enter balance:";
 cin >> balance;
  }
 };
 int main()
 {
 account a[10];
 for (int i=0; i<10; i++)
 {
cout <<"\n Enter details for customer " << i+1 <<"
    \n;
```

Page No.:
Date:

Details for customer1:
name: oliver
account_no: 12456789
balance: 78000

Details for customer2:
name: Vedant
account_no: 87848245
balance: 2400

Details for customer3:
name: pushkar
account_no: 97458452
balance: 5214

Details for customer4:
name: Ayush
account_no: 78211243
balance: 21211

Details for customer5:
name: Mayuresh
account_no: 94512617
balance: 1112

Details for customer6:
name: yuvraj
account no: 12314587
balance: 2999

Details of customers 7:
name: Zack
account_no: 124 789 754
balance: 12478

Details of customers 8:
name: Jack
account_no: 343 343 419
balance: 31121

Details of customers 9:
name: Joye
account_no: 992 447 989
balance: 90000

Details of customers 10:
name: Sam
account_no: 134 679 a74
balance: 200000

3] Write a program to declare a class 'staff' having data members as name & post. Accept this data for 5 staff & display names of staff who are "HOD".

—>

```
#include <iostream>
#include <strings>
  using namespace std;
  class staff
  {
   char name;
   char post;
   public:
   void accept()
   {
  cout<<"Enter name:";
  cin>>name;
  cout<<"Enter Post:";
  cin>>post;
   }
   void display()
   {
  if(strcmp(post,"HOD")==0)
   {
  cout<<"HOD name"<<name<<endl;
    }
   }
  };
 int main()
 {
  staff s[5];
 for(int i=0; i<5; i++)
```

```
{
cout<<" List details for staff "<< i+1 ":"
<< endl;
s[i].accept();
}
cout<<"\n List of staff with post 'HOD':"
<< endl;
for(int i=0; i<5; i++)
{
s[i].displayifHOD();
}
return 0;
}
```

* Output

```
Details for staff 1:
  Name: James
  Post: HOD

Details for staff 2:
  Name: Charles
  Post: Lecturer

Details for staff 3:
  Name: Henry
  Post: Principle

Details for staff 4:
  Name: william
  Post: HOD

Details for staff 5:
  Name: oliver
  Post: Director

List of HOD's =
Name: James
Name: William.
```

26/8

Experiment 3

Page No.:
Date:

1] Write a program to class book - data members as book - title, author - name & price. Accept & display into for 1 object using this pointer.

→

```
#include <iostream>
using namespace std;
class book {
private:
string b-tittle, a-name;
int price;
public:
void accept () {
cout <<" Enter book tittle = ";
getline (cin, this -> b_tittle);
cout <<" Enter author name = ";
getline (cin, this -> a_name);
cout <<" Enter price = ";
cout <<" Enter price = ";
cin >> Price;
}
void display () {
cout <<" The book " <<b_tittle <<" author
name: " <<a_name <<" & price = "
<< Price << endl;
}
} b1;
int main () {
    book * a;
    a = & b1;
    a -> accept ();
    a -> display ();
}
```

* Output
Enter the book title = Apple
Enter the author name = Einsten
Enter the price = 151
The book Apple. Author name Einsten & price = 15

Q.2] // Write a program to class student, data members roll_no, percent. Accept & display using this pointer for 1 object.

→

```
#include <iostream>
using namespace std;
class student {
public:
int roll_no;
float percent;
void accept () {
cout << "Enter roll no. & percentage = ";
cin >> this -> roll_no >> this -> percent;
}
void display () {
this -> Accept ();
cout << " Roll no. = " << this -> roll_no;
cout << "Percentage = " << this -> percent;
}
};
int main () {
student s1;
s1. display ();
return 0;
}
```

Page No.:
Date:

* Output
Enter roll no. & percentage = 12 92.7
Roll no. = 12
Percentage = 92.7

3] Write a program to demonstrate the use of nested class:

→

```
#include <iostream>
using namespace std;
class demo {
public:
int roll_no;
string name;
void accept () {
cout << "Enter the name of student & roll no. =
cin >> name >> roll_no;
void display () {
cout << "\n Name = " << name << "\t roll_no =
    << roll_no << endl;
    }
  }
};
int main () {
demo :: dem2 s1;
s1.accept ();
s1. display ();
return 0;
}
```

2818

* Output
Enter name of student & roll_no.
Apple 12
Name = Apple
Rollno. 14

Experiment - 4

1] WAP to swap two numbers from same class using object as function arguments. Write swap function as member function.

→

```
#include <iostream>
using namespace std;
class Number {
 int num;
 public:
 void accept ()
 {
 cout << "Enter number: ";
 cin >> num;
 }
 void display () {
 cout << "Number: " << num << endl;
  }
 void swap (Number & obj) {
 int temp = num;
 num = obj.num;
 obj.num = temp;
 }
};
int main() {
 Number n1, n2;
cout << "Enter first number : " <<endl;
 n1.accept ();
cout << "Enter second number : " <<endl;
 n2.accept();
 n1.swap(n2);
 cout << "\n After swap. " << endl;
 cout << "First";
 n1.display ();
```

```
cout <<" second" : ;
n2. display ();
 return 0;
 }
```

* Output

Enter first no.
Enter number: 6
Enter second number:
Enter number: 9
After swap:
first number: 9
second number: 6.

2] WAP to swap two numbers from class using concept of friend function.

→

```
#include <iostream>
 using namespace std;
  class temp {
  int x, y, q;
  public:
   void accept ()
   {
cout <<" Enter two numbers ";
cin >> x >> y;
  }
Void display()
 {
cout<<" After swap x is : " <<x;
cout<<" After swap y is : " <<y;
  }
```

```
friend void swap(temp &t);
};
void swap(temp &t)
{
t.q = t.x;
t.x = t.y;
t.y = t.q;
}
int main()
{
temp t1;
t1.accept();
swap(t1);
t1.display();
return 0;
}
```

* Output

```
Enter two numbers
6

After swap x is: 6
After swap y is: 9
```

3) WAP to swap two numbers from different class using concept of friend function.

→

```
#include <iostream>
using namespace std;
class B;
class A;{
int num A;
public:
 void accept () {
cout<<"Enter number A:" ;
cin >> num A;
 }
 void display();
cout <<" Number A ="<<num A<<endl;
 }
friend void swap number (A&, B&);
 };
 class B {
  int num B;
  public:
  void accept () {
 cout <<"Enter number B :";
 cin>> numB;
  }
  void display () {
 cout<<" Number A = "<< numB << endl;
 }
Friend void swap numbers (A&, B&);
 };
void swap numbers :(A&a, B&b) {
```

```
int temp = a.numA;
 a.numA = b.numB;
 b.numB = temp;
}
int main () {
 A c1;
 B d1;
c1.accept();
d1.accept();
swapnumbers (c1, d1);
cout<<"\n After swaping :"<<endl;
c1.display();
d1.display();
return 0;
 }
```

* Output

Enter number A: 62
Enter number B: 72

After swapping

Number A = 72
Number B = 62

4) WAP to create two classes result1 & result2 which search the marks of students, read the value of a marks for both class object & compute the avg of 2 results.

→

```
#include <iostream>
using namespace std;
class result2;
class result1;
 int a;
 public:
 void accept(){
cout<<" Enter marks out of 50";
cin>> a;
 }
friend void cal(result1, result2);
 };
 class result2 {
  int b;
  public:
  void accept (){
cout<<"Enter marks
cout<<"Enter marks out of 50:";
cin >> b;
 }
friend void cal(result1 , result2 ){
 };
void cal (result1 r1, result2 r2) {
  float avg = (float) (r1.a + r2.b);
cout<<"\n average : "<<avg;
 }
int main()
 {
```

```
result1 x;
result2 y;
x.accept();
y.accept();
cal(x,y);
}
```

* Output

```
Enter marks out of 50 : 45
Enter marks out of 50 : 47
Average = 45.6
```

5) WAP to find the greatest number among 2 numbers from two different classes using friend function.

→

```
#include<iostream>
using namespace std;
class B;
class A {
int a;
public:
void accept() {
cout<<" Enter a value :";
cin>>a;
}
friend void greater (A a1, B b1);
};
class B {
int b;
public:
void accept() {
cout<<" Enter a value";
```

```
    cin>>b;
    }
friend void greater (A a1, B b1);
    };
void greater (A a1, B b1) {
      if (a1.a > b1.b) {
cout<<" First value is greater";
    }
    else {
cout<<" Second value is greater";
    }
    }
    int main()
    {
    A x;
    B y;
     x.accept1();
     y.accept1();
     greater (x, y);
     }
```

Q 28/5 practice Q. ?

Experiment - 5

a) WAP to find the sum of numbers 1 to n, where n is a variable passed to constructor.

→
```
#include<iostream>
using namespace std;
class w {
  int n;
  public:
  w() { // Default constructor
  int i, sum=0;
    n=10;
    for(i=1; i<=n; i++) {
     sum+=i;
    }
  cout<<"The sum is "<<sum<<endl;
  }
  w(int num) { // Parameterized constructor
  int i, sum=0;
  for(i=1; i<=n; i++) {
  sum+=i;
  }
  cout<<"The sum is "<<sum<<endl;
  }
  w(w& obj1) { // Copy constructor
  n=obj1.n;
  int i, sum=0;
  for(i=1; i<=n; i++) {
  sum+=i;
  }
  cout<<"The sum is "<<sum<<endl;
  }
};
```

```
int main() {
 w obj1; //Default constructor invocation
 w obj2(5); // Parameterized constructor invocation
 w obj      // Copy constructor invocation
 return 0;
 }
```

b) Write a program to declare a class "student" having data members name & percentage. Write constructor to intialize these data members & accept & display for 1 student.

→
```
#include <iostream>
 using namespace std;
 class student {
 string name;
 float percentage;
 public:
 student () {
 name = "John Doe";
 percentage = 98.5;
cout << "Name: " <<name"; " <<percentage "<<%.<<end
 }
student (string n, float p) {
 percentage = p;
 name = n;
cout <<"Name: " <<name<<", " <<percentage <<%.<<e
 }
 student (student & obj1) {
 percentage = obj1.percentage;
 name = obj1.name;
cout<<"Name:" <<name <<", "<<percentage<<%.<<endl;
 }
 };
```

```
int main () {
 student s1;
 student s2("John Joe", 47.9);
 student s3(student s2);
 return 0;
 }
```

* o/p

Name: John Doe, 98.4%.
Name: John Joe, 47.9%
Name: John Joe, 47.9%.

c) Define class "college" data members as roll no, name, course. Accept & display data for 2 students

→

```
#include <iostream>
 using namespace std;
 class college
 {
 int roll;
 string name, course;
 public:
 college (int r, string n)
 {
  roll r;
 name n;
 course = " computer science";
 }
 void display ()
  {
 cout<<" The name of student = " <<name <<endl;
 cout <<" The roll number of student = " <<roll <<endl
 cout <<" The course of student = " <<course <<endl;
  }
 };
```

```
int main()
{
cout<< "The name of student = "<<name<< endl;
cout<< The roll number of stud.
college. s1 (11, "Siddhant") s2(10, "Ameya");
s1. display ();
s2. display ();
}
```

* O/P

The name of student = Siddhant
The roll number of student = 11
The course of student = Computer science
The name of student = Ameya
The roll number of student = 10
The course of student = computer science.

d) Constructor over loading

```
#include <iostream>
using namespace std;
class Rectangle ;
{
 int l, w;
 public:
rectangle ( ) {
 l=1;
 w=2;
}
rectangle (int a)
{
 l=a;
 w=b;
}
rectangle (inta, int b)
{
  l = a;
  w = b
}
void calculate ()
{
int a;
a=lxb;
cout <<"Area = " <<a;
}
};
```

```
int main()
{
rectangle r1;
rectangle r2 (5);
rectangle r3 (4,5);
r1. calculate ();
r2 . calculate ();
r3 . calculate ();
}
```

20/9/25

Experiment - 6

a) Single inheritance

→
```
#include <iostream>
using namespace std;
class person{
protected;
string name;
int age;
};
class derived: protected person {
        private:
        int roll;
        public:
        void accept() {
cout<<"enter name, roll no. & age = ";
cin >> name >> roll >> age;
}
void display() {
cout<<endl << "Name = " << name<<endl <<
"Roll no. = " <<roll << endl <<"Age = " << age
<<endl; }
};
int main() {
derived d1;
d1.accept();
d1.display();
return 0;
}
```

b] Multiple inheritance

```
#include <iostream>
using namespace std;
class academic {
    public:
    int marks;
    };
  class sports {
     protected:
     int score;
     };
class result: protected academic, protected.
      sports {
      int total;
      public:
      void accept () {
cout<<"enter clg marks & sports score=";
cin >>marks >> score;
 }
 void disp() { cal. ();
cout <<" The total marks = " << total;
 }
 int main () {
   result. rl;
   rl. accept ();
   rl. display ();
   return 0;
   }
```

c] Multi-level inheritance

→
```
#include <iostream>
using namespace std;
class vehicle {
    public:
    string brand, model;
    };
    class car: public vehicle {
    public:
    string attribute;
    };
    class ecar: protected car {
    private:
    string battery = "capacity";
    public:
    void accept () {
cout << "enter brand car type, model, battery,
    capacity = "<<; << model;
cin >> brand >> attribute >> battery - capacity;
  }
  void display () {
cout << endl << "car brand = " << brand << end <<
  "car type = " << attribute << endl << "car model ="
 << model << endl << "car_battery = " << battery-
  capacity << endl;
  }
};
  int main() {
  ecar el;
  el.accept ();
  el.display();
  return 0;
  }
```

d) Heirarchical inheritance

```
#include <iostream>
using namespace std;
class employee {
  public:
  string id_name;
  };
class manage : protected employee {
   private:
   string id_name; dept;
    public:
    void acc () {
cout<<" enter id_name ; dept = ";
cin >> id >> name >> dept;
  }
  void disp () {
cout<<" emp_id = "<<id <<" \n name :"
   <<name <<" \n emp dept :" <<dept<<endl;
  }
};
class developer : protected employee {
      private:
string programing_language = " Rust ";
  public:
  void disp () {
cout <<" Programing language :" << programing
      language <<endl;
   }
  };
```

```
int main () {
manager m1;
developer d1;
m1. acc();
m1. disp();
d1. disp();
return 0;
}
```

e) Hybrid inheritance

```
#include <iostream>
using namespace std;
class person {
  protected;
  string name;
  int age;
};
class sports {
 public:
 int score;
 void acc() {
 cout << " enter sport score = ";
 cin >> score;
}
};
class academic {
   public:
   int marks;
   void acc () {
cout << " enter marks of student = ";
cin >> marks;
}
};
```

```
class student: protected person {
 public:
 void accept () {
 cout <<" enter age & name of std = ";
 cin >> age >> name;
 }
};
class result : protected sports protected
student, protected academic {
  public:
  int total;
  void disp() {
   student:: accept();
  academic:: accept();
  sports :: accept();
  }
void cal () {
total = marks + score;
cout <<endl<<" total marks of students = "<<
    total <<endl;
  }
  };

  int main() {
   result r1;
   r1.disp();
  r1.cal();
  return 0;
  }
```

f) Virtual base class

```
#include <iostream>
using namespace std;
class college {
 protected:
  int id;
  string name;
 };
class test: virtual protected college {
   public:
    int per;
    void acc() {
cout << endl <<" enter student id & name =";
cin >> id >> name;
 }
void acc () {
cout <<endl <<" enter student marks =";
cin >>per;
 }
};
class sports : virtual protected college {
   public:
   char grade;
    void acc() {
 cout <<endl <<" enter grade=";
  cin >> grade;
  }
 };
class result : protected test, protected sports
   public:
   void disp () {
```

```
test::acc();
test::acc();
sports::acc();
}
};
int main() {
    result r;
    r.disp();
    return 0;
}
```

Q

29/9/25

Experiment - 7

a) WAP using function overloading to calculate the area of a square classname a rectangle laboratory.

→
```
#include <iostream>
using namespace std;
class b {
    public:
    void area (int a) {
    cout <<" The area is "<< a*a <<endl;
    }
    void area (int a, int b) {
    cout<< " The area is "<<a*b<<endl;
    }
} b1;
int main() {
d.area(14);
d.area(10,30);
 return 0;
}
```

* O/P

The area is 196
The area is 300

b) WAP using function overloading to calculate the sum of 5 float values & 10 integers

→
```
#include <iostream>
using namespace std;
class c {
    public:
void sum (in arr [], int size) {
  int i; sum = 0;
for (i=0; i<size; i++) {
```

```
cout << " Sum is " << sum << endl;
  }
void sum(float arr[], int size) {
    int i;
    float sum = 0;
    for( i=0; i < size ; i++) {
        sum += arr[i];
    }
cout << " Sum is " << sum << endl;
  }
} d;
int main() {
int arr[10] = {10, 20, 30, 40, 50, 50, 40, 30, 20, 10};
float arr2[5] = {13.7, 12.6, 17.9, 19.8, 15.5};
 cl.sum(arr, 10);
 cl.sum(arr2, 5);
  return 0;
```

* O/P

Sum is 300

Sum is 79.5

c) WAP to implement '-' operator (unary) when used the numeric data is negated.

→

```
#include <iostream>
  using namespace std;
  class d {
    int a;
    public:
    void operator - () {
        a = -a;
    }
```

```
    void getdata() {
cout<<" Enter Number: " ;
cin >>a
 }
void showdata () {
 cout<<"New value: "<<a;
  }
} dl;
 int main () {
  dl.accept ();
  -dl;
  dl. display ();
  }
```

* O/P

Enter number: 17

New value: -17

d) WAP to implement '++' operator (unary) so when used the numeric data is increment.

```
#include <iostream>
 using namespace std;
  class e {
     int a;
     public:
     void accept () {
    cout<<" Enter number: ";
    cin >>a;
     }
   void display () {
  cout <<"New value: "<<a;
  }
```

```
friend void operator ++ (e & a);
friend void operator ++ (e & a, int);
 } e1;
void operator ++ (e & a) {
    ++ a.a;
   }
void operator ++ (e & a, int ) {
    a.a++;
  }
int main() {
   e1.accept();
   ++ e1;
   e1.display();
   e1++;
    return 0;
     }
```

* O/P

```
Enter number: 137
New value : 138
New value : 139
```

Experiment. 8

Page No.:
Date:

a] WAP to overload the binary + operator th that two strings are concatenated.

```
#include <iostream>
using namespace std;
class w {
  string x;
  public:
  void getdata() {
  cout << " Enter String : ";
  getline ( cin, a);
   }
  string operator + ( w & x) {
     return   x + x.x;
   }
 } w1, w2;
  int main () {
    w1. getdata ();
    w2. getdata();
  cout << " Concatenated string = " << w1 + w2;
   return 0;
   }
```

* O/P

```
Enter string : Siddhant
Enter string : Mane
Concanteenated string : Siddhant Mane.
```

b) WAP to create base class I login with data members Name & Password. Declare getdata() as a virtual. Derive Email login & Membership Login classes from I login. Display Email login details & membership Login details of the employee.

→
```
#include <iostream>
using namespace std;
class I Login {
protected:
string name, pwd;
virtual void getdata() = 0;
};
class Email login: protected I login {
    string Email;
    public:
    void get data() {
    cout << "Name : ";
    getline (cin, Name);
    cout << "Email : ";
    getline (cin, Email);
    cout << "Password : ";
    getline (cin, pwd);
    }
Void show data () {
cout << "\n Email login \n Name : " << Name << Email:";
Emai << "\n Password : " << pwd << endl;
}
};
```

```
class Membership Login : protected ILogin {
    string MID;
    public:
    void getdata() {
    cout<<"Name: ";
    getline (cin, MID);
    cout<<"Password: ";
    getline (cin, pwd);
    }

  void showdata() {
  cout<<"\n Membership Login \n Name : " <<Name
   <<"\n Membership ID: " << MID "\n<<\n
    Password: "<< pwd <<endl;
     }
  } x1;
   int main () {
   cout<<"Enter Email Login : "<<endl;
   x.getdata();
   x.showdata();
   x1.showdata();
     return 0;
   }
```

* O/P

Enter Email login details:
Name: Siddhant Mane
Email: Siddhant.Mane@gmail.com
Password: xyz@123

Email Login
Name: Santosh kute
Email: Santosh@gmail.com
Password: abc@146

12/11

Experiment - 9

Page No.:
Date:

a) WAP to copy the contents of one file to another open the first file in read & second file in out mode. copy contents of first file in second file. Assume second file is already

→
```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
if_stream one;
one.open ("first.txt");
of stream two;
two.open ("second.txt);
if (!one || !two) { cout<<"Error"; return-1;}
char ch;
3
cout<<"Text copied successfully!";
one.close();
two.close();
return 0;
3
```

* O/P
Text copied successfully

(b)(c)(d) WAP to count digits, spaces & ocurences of word in a file.

→
```
#include<iostream>
#include<fstream>  ← #include<cctype>
using namespace std;
int main() {
ifstream tank;
tonk.open ("test.txt");
```

```
if (!tonk) {
cout<<" Error. file corrupted or non-exist";
return -1;
}
char ch;
int space=0, digit=0, wcount=0 cwcount=0;
string word ; w;
cout<<"Enter a word which occurence must be
    counted" << endl;
    cin>> word;
    if(isdigit (ch)) {
     digit ++;
    }
   else if (ispunct (ch) {};
   else if (isalpha (ch)) {
     w += ch;
    }
   if (isspace (ch)) {
    space ++;
    if(w!= " ") {
     wcount ++;
     if (w== word) {
       cwcount ++;
     }
    }
    w= " ";
    }
  if (w = " ") {
   wcount ++;
   if(w== word) {
   cwcount ++;
  }
}
```

```
cout << "Spaces: " << space << " Digits: " << digit
<< " word: " << wcount << " Occurence of " <<
word << " : " << cwCount << endl;
return 0;
```

* O/P

Spaces: 17 Digits: 31 , words: 18 Occurrences of banana: 1

12/11

Experiment - 10

a) Sum of array elements using function template

```
#include <iostream>
using namespace std;
template <typename>
T sumArray (T arr[], int size);
    Tsum = 0;
for(int i= 0; i<size; i++) {
    sum t = arr[i];
    }
return sum;
    }
int main() {
const int size = 10;
int Array[size] = {1,2,3,4,5,6,7,8,9,10};
int intsum = sum array (int Array, Size);
cout<<"Sum of integer array: "<<int sum;
return 0;
}
```

*o/p

55

59.6

b) Square Function (Template Specilization)

```
#include <iostream>
#include <string>
using namespace std;
template <typename T>
T square (T value) {
return value * value;
}
template <>
```

```
string square<string> (string str) {
 return str+ str;
 }
 int main () {
 int intNum = 5;
cout<<"Square of integer"<< intNum <<":"
   <<square (num)
string mystr = "Hello";
cout<<"Square of string"<<"mystr"
<<":" << square (By Sh I)
 return 0;
 }
```

* O/P

25

Aaa Aaa

c) Simple Calculator (use template)

→

```
#include <iostream>
#include<string>
 using namespace std;
 template <typename T>
  class calculator {
  private:
  T num1;
  T num2;
  public:
calculator (T n1, T n2): num (n1), num (n2)
T add (.) const {
 return num1 - num2;
 }
T substract() const {
  return num1 - num2;
  }
```

```
T multiply () const {
  return num1 * num2;
  }
T divide ()  const {
  return num1/num2;
  return 0;
  }
```

d) Stack implementation (class template)

→
```
#include <iostream>
#include <vector>
  using namespace std;
  class template <typename T>
  class stack {
  private:
  vector <T> elements;
  public:
  void push (T val) {
  elements.push.back (val);
  cout<<" Pushed :" <<val;
  }
 T.pop ()
  {
 ;f (elements empty ()) {
 cout <<' 'Error: stack is empty "<<endl;
  return T();
 T top element = elements.back ();
 elements.pop_back ();
  return top Element;
  }
 cout is empty () const {
  return element.empty ()
  }
 }; return 0; }
```

Experiment - 11

Page No.:
Date:

return 0; WAP to implement generic vector.

a) To modify the value of given element.

```
#include<iostream>
template <typename T>
class vector {
 T a[100];
 int size;
 public:
vector (int s) : size(s) { }
void set (int i ; T val) {
 if ( i >=0 && i<size)
 a[i] = val;
  else
cout << " invalid ";
  }
T get (int i) {
 if ( i>=0 && i<size)
  return a[i];
  cout << " Invalid ";
  return T;
  }
void display () {
 for ( int i=0 ; i<size ; i++ )
  cout << a[i] << " ";
  cout <<endl;
  } };
int main() {
 vector <int> v(5);
  for( int i=0 ; i<5 ; i++) {
   v.set ( i, i*10);
   v. display ();
   v.set (2,aa);
```

```
cout<<" After modification ";
v. display();
return 0; }
```

* O/P

0 10 20 30 40

After modification : 0 10 99 30 40

b) To multiply by a scalar value

—>
```
#include <iostream>
#include <vector>
    using namespace std;
     int main() {
    vector <int> vec = {1, 2, 3, 4, 5};
    int scalar = 3;
    for (int, value : vec) {
    val = val * scalar; }
    for (int & value : vec) {
    cout<< val<<" "; }
    cout<<endl;
     return 0;
      }
```

* O/P

3 6 9 12 15

c) To display the vector is the form (10, 20, 30)

→
```
#include <iostream>
#include <vector>
  using namespace std;
  int main() {
vector <int> vec = {10, 20, 30, 40, 50}
cout << "(";
for (int i=0; i<vec.size() ; i++) {
cout << vec[];
if (i != vec.size() - 1)
cout << " ";
  }
cout << ")" << endl;
  return 0;
```

*O/P

(10, 20, 30, 40, 50)

12/11

Experiment - 12

Page No.:
Date:

Q.] WAP using STL

a) Implement Stack

→
```
#include <iostream>
#include <stack>
using namespace std;
int main() {
stack <int> s;
s.push(10);
s.push(20);
s.push(30);
cout<<" Top element : " << s.top() <<endl;
s.pop();
cout<<" Top element after pop : << s.top()<<e
cout<<" Stack size : " << s *size() <<endl;
if (s.empty()) {
cout<<" Stack is empty "<<endl;
} else {
cout<<" stack is not empty "<<endl;
}
return 0;
}
```

* O/P

```
Top element: 30
Top element after pop : 20
stack size : 2
stack is not empty
```

b) Implement queue

→
```
#include<iostream>
#include<queue>
using namespace std;
int main() {
queue<int> q;
q.push(10);
q.pop(20);
q.push(30);
cout<<"front elements:"<<q.front()<<endl;
cout<<"back elements:"<<q.back()<<endl;
q.pop()
cout<<"after pop operations"<<endl;
cout<<"queue.size"<<q.size()<<endl;
if(q.empty) {
cout<<"Queue is empty:"<<endl;
} else {
cout<<"Queue is not empty:"<<endl;
}
return 0;
}
```

* o/p

```
Front element: 10
back element: 30
after pop;
front element: 20
queue: 2
queue is not empty
```

c)

→
```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
struct person {
string name;
int age;
Person(string n, int a): name(n), age(a){
};
int compare_age(const person& p1, const person
& p2);
return(p1.age< p2.age)? 1:0; }
int main() {
vector < person > person = {
person("Alice", 30);
person("Bob", 25);
person("Charl", 35);
person("Dave", 28); };
Sort(people begin(), people.end());
[&](const person& a. const Person & 6) {
return (compare-age (a, b); }
cout<< "Sorted records by age: \n";
for (auto & p: people) {
cout << p: name " - " << p.age << "\n";
int sample age = 28;
int found index = -1;
for(int i:= 0; i< (int) people.size(); i++) {
if(people [i].age =search_age? 1:0) {
found index = i;
break; } }
if (found != index)
```

```
cout << " person with age" << search-age << endl;
else
cout << " person with age" << search-age << "
        not found" ;

    return 0;
    }
```

O/P

Sorted records Age:

Bob = 25

Dave = 28

Alice = 30

Charl = 35