Q.2 write c++ program to peryorm arithmatic operations using switch case.

→

```
# include <iostream.h>
using nameplace std;
int main()
{
cout<< "ARITHEMATIC OPERATIONS";
cout<<
double num1, num2;
char op;C
cout << "Enter two numbers";
cin >> num1 >> num2;
cout << "Enter an operation";
Switch (op) {
case '+':C
cout<< "Result: " << num1+num2;<<endL;
break;
case '-':
cout << "Result: "<< num1-num2;<<endL;
break;
case '*':
cout<< "Resut: "<< num1* num2;<<endL;
break;
case '/':
if (num2 != 0)
cout<< "Result: "<< num1/num2<<endL;
<< endL;
else
cout<< "Error";
break;
```

```
case "%":
if (num2 != 0)
cout << "Result : " << num1 % num2 << endl;
else
cout << "Error" << endl;
break;

default:
 cout << "Invalid operator!" << endl;
}
return 0;
}
```

(SYNTAX)

```
switch (expression) {
case (value1) :
 " code
break;
....
```

OUTPUT :-

```
Enter 2 numbers: 3  4
Enter an operation: +
Result: 7
```

Q.3 program to check wheather a numbers is even or odd.

→

```
# include<iostream>
using namespace std;
int main()
{
int a;
cout<< "enter a number a: ";
cin>>a;
if (a % 2 == 0)
{
cout << "the number "<<a<<" is even.";
}
else
{
 cout<<" the number "<<a<<" is odd";
}
return 0;
}
```

OUTPUT :-

```
enter a number a: 4
the number 4 is even
```

Q.4 c++ code to printf 1-10 numbers using loop.

→

```
# include <iostream>
using namespace std;
int main()
 {
  int i;
 for(i=1; i<=10; i++)
 cout <<i<<endl;
 return 0;
 }
```

output :-

1
2
3
4
5
6
7
8
9
10.

Q.5 C++ code to print 10 - 1 using while loop

→
```
#include <iostream>
using namespace std;
int main()
{
  int i=10;
  while (i>=1)
  {
   cout<<i<<endl;
   i--;
  }
  return 0;
}
```

output :-

10
9
8
7
6
5
4
3
2
1

Q.6 C++ code to printf

(a)
```
  *
 * *
* * *
```

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

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

→
```
#include <iostream>
using namespace std;
int main(){
  int i,j;
  for(i=1;i<=5;i++)
  {
    for(j=1;j<=i;j++)
```

```
    {
        cout<<i;
    }
    cout<<endl;
    }
    return 0;
}
```

(c)
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

→
```
#include <iostream>
using namespace std;
int main() {
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1;j<=i;j++)
        {
            cout<<j;
        }
        cout<<endl;
    }
    return 0;
}
```

8/7/25

Experiment 1

1) WAP to declare a class student having data members as RollNo. & Name. Accept and display data for single S.

→
```
#include <iostream>
#include
using namespace std;
class student
{
    int Roll_no;
    string name;
    public:
    void accept()
{
    cout<<" Enter name and RollNo:" ;
    cin >> name >> Roll_no;
}
    void display()
    cout<< "Name:"<< name <<endl;
};

int main()
{
    student S1;
    S1.accept();
    S1.display();

return 0;
}
```

output:-
Enter name and Roll no: sanika
1
Name: sanika
Roll no: 1

Q.2 WAP to create a class book having data members as book_name, b_price, b_pages and display the name of book having greater price.

→
```
#include<iostream>
using namespace std;
class book
{
 public:
 string b_name;
 int b_pages;
 int b_price;
void accept()
{
 cout<<"Enter book name, no. of
        Pages and price:";
 cin>>b_name>>b_pages>>b_price;
}
void display()
{
 cout<<"enter book name, no. of Pages
        and price:";
 cin>>b_name>>b_pages>>b_price;
}
```

```
int main()
{
 book B1, B2;
 B1.accept();
 B2.accept();
 if (B1.b_price>B2.b_price)
{
    B1.display();
}
 else
{
    B2.display();
}
    return 0;
}
```

Output:-
enter book name, no. of pagess and price: alchemist 45 500
enter book name, no. of pages and price:
youngminds
235
7000
book name: youngminds
book price: 700
no. of pages: 235

Q3

```
→ class timesec
{
   int h, m, s;
   char c;
   Public:
    void accept()
{
   cout<<" Enter time in HH:MM:SS
              format: ";
    cin>>h>>c>>m>>c>>s;
}
  void display() {
    int totalseconds = h*8600 + m* 60
                    + s;

    cout << " Time in seconds :" <<
         totalseconds << endl;
}
};
int main()
  timesec t;
  t.accept();
  ptr -> accept c
  return 0;
}
```

output:-
enter time in HH·MM·SS format
20:44:21

Time in seconds : 74661

EXPERIMENT 2

a) WAP to declare a class 'city' having data members as name and population. Accept this 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 population:";
    cin>>population;
}
  void display(){
    cout<<"city:"<<name<<",population
    :"<<population<<endl;
}
};
int main(){
    city c[5];
    int i, maxindex=0;
   for(i=0;i<5;i++){
     cout<<"city"<<i+1<<":"<<endl;
     c[i].accept();
   }
 for(i=1;i<5;i++){
    if(c[i].population>c[maxindex].
population){
    maxindex=i;
}
}
  cout<<"In city with the highest
        population:\n";
    c[maxindex].display();
    return 0;
}
```

Output:-

```
city 1:
enter city name: p
enter population: 45
city 2:
        "    : b
        "    : 678
city 3:
        "    : 9
        "    : 567
city 4:
        "    : d
        "    : 90
city 5:
        "    : w
        "    : 566
```

city with highest population:
city: b , population: 678

(6) WAP to declare a 'class' 'Account' having data members as account no. and balance. Accept this data for 10 accounts and give interests of 10% where balance is equal or greatere than 5000 and display.

→
```
#include<iostream>
using namespace std;
class account{
Public:
    int accNo;
    float balance;
    void accept(){
    cout<<"enter account number:";
    cin>>accNo.;
    cout<<"enter balance:";
    cin>>balance;
}
Void giveintrest(){
    if(balance>=5000){
        balance+=balance*0.10;
    }
}
Void display(){
    if(balance>=5000){
    cout<<"Account NO:"<<accNO
    <<",updated balance:"
    <<balance<<endl;
}
};
```

```
int main(){
    Account a[10];
    for(int i=0;i<10;i++)
    {
        cout<<"Account"<<i+1<<":";
        a[i].accept();
    }
    cout<<"updated:";
    for(int i=0;i<10;i++){
        a[i].giveintrest();
        a[i].display();
    }
    return 0;
}
```

Output :-

```
Account 1:
enter account number: 1
enter balance: 1000
Account 2:
            : 2
            : 2000
Account 3:
            : 3
            : 3000
Account 4:
            : 4
            : 4000
Account 5:
            : 5
            : 5000
            ..... etc.
```

updated:
Account No: 5, updated Balance: 5500
(so on) .....

(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{
public:
 string name;
 string post;
 void accept() {
 cout<<"enter staff name:";
 getline(cin, name);
 cout<<"enter post:";
 getline(cin, post);
}
 void displayIfHOD() {
  if (post=="HOD"||post=="hod"||
      post=="HOd") {
   cout<<name<<" is HOD";
  }
 }
};
int main(){
 staff s[5];
 cin.ignore();
 for(int i=0; i<5; i++){
  cout<<"staff"<<i+1<<":\n";
  s[i].accept();
 }
 cout<<"\n list of staff who are HOD:\n";
 for(int i=0; i<5; i++) {
  s[i].displayIfHOD();
 }
  return 0;
}
```

output :-
staff 1:
enter staff name: sanika
enter post: HOD
staff 2:
enter staff name: hrishi
enter post: CEO
.....
List of staff who are HOD:
spruha is HOD
anushka is HOD.

29/7/25

Exp 3

Q.1 WAP to declare a class 'book' containing data members as book title & author_name.

→
```
#include <iostream>
#include <string.h>
using namespace std;
class Book
{
  private:
    string book_Title;
    string author_Name;
    float price;
  Public:
  void accept()
{
  cout << " enter book title: ";
  getline (cin, book title);
  cout << " Enter author name: ";
  cin >> autor_name;
  cout << " enter price: ";
  cin >> price;
}
void display()
{
  cout << " Book title: " << book-title
  << endl;
  cout << "Author name: " <<
      author_name << endl;
  cout << "Price: " << price << endl;
}
return 0;
}
```

b) WAP to declare a class 'student' having data members roll_no & percentage using 'this pointer' invoke member function to accept & and display for 1.

→
```
#Include <iostream>
using namespace std;
class student {
Public:
  int roll no;
  float percentage;
  void accept() {
  cout << " enter roll no: ";
  cin >> roll no;
  cout << " enter percentage: ";
  cin >> percentage;
}
  void display()
{
    cout << "In student INFO\n";
    cout << "roll no: " << roll no <<
                                endl;
    cout << " percentage: " << percentage
}
};
int main() {
  student s1;
  s1.accept();
  s1.display();
  return 0;
}
```

OUTPUT :-
enter roll no : 1
enter percentage : 90

STUDENT INFO
roll no. : 1
percentage : 90

Exp 4

Q.1 WAP to swap 2 numbers from same class using object as functions argument write swap function as member function

→
```
#include <iostream>
using namespace std;
class Number {
    int value;
    Public:
    void getvalue() {
    cout << "enter value :";
    cin >> value;
}
void display() {
   cout << " value << endl;
}
void swapValues (Numbers &obj ) {
    int temp = values;
     valence = obj.value;
     Obj.value = temp;
}
};
   int main() {
   Number n1, n2;
      cout << "enter the first value
                of object:";
        n1, n2 ;
   cout << " enter first value fo
```

```
    object :";
n1 . getvalue ();
cout << "enter value for second
        object :";
n2 . getvalue ();

cout<< "In Before swapping "<<endl;
cout<< "n1 = "; n1 . display();
cout<< "n2 = "; n2 . display();
 n1 . swapvalues (n2);

cout << " After swapping :"<<endl;
cout << "n1 = ". n1 . display();
cout << "n2 = "; n2 . display();
return 0;
}
```

O/P: Enter value for first object :
enter value : 5
enter value for second object :
enter value : 8
Before swapping : $n_1 = 5$
$n_2 = 8$

After swapping : $n_1 = 8$
$n_2 = 5$

Q.2 WAP swap 2 numbers from same class using concept of friend functions

→

```
#include <iostream>
using namespace std;
class Number {
private :
 int n1, n2;
Public :
 void accept()
{
    cout << "enter two numbers";
    cin >> n1 >> n2;
}
void display()
{
    cout << " n1 = " << n1 << endl;
    cout << " n2 = " << n2;
}
friend void swapNumbers (Number &n)
    int temp = n.n1;
        n.n1 = n.n2;
        n. n2 = temp;
}
};
int main () {
    Number m;
    m.accept();
 cout << " Before swap :";
    m. display();
swap (m);
 cout << " After swap :";
```

```
    m.display();
    return 0;
}
```

O/P:- Enter 2 numbers : 2
3
Before swap: $n_1 = 2$ 4
$n_2 = 3$
After swap : $n_1 = 3$
$n_2 = 2$

Q.3 WAP to swap 2 numbers from diff. classes using friend function.

→

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

class class 2;
class class 1;
private :
int num1;
Public :
void accept () {
cout << " enter first number :";
cin >> num1;
}
void display () {
cout << "class value :" <<
num1 << endl;
}
friend void swapNumbers (class1 &
                         as class 2&)
};
```

```
class class 2 {
private :
  int num 2;
Public :
void accept () {
  Cout << "enter second number :" ;
  Cin >> num 2;
}
void display () {
  Cout << "class 2value :" << num 2
  << end ;
}
friend void swapNumbers (class 1 & a
                         class 2 & b) ;
};
void swapNumber ((class1 & a . class
                  2 & b) {
    int temp = a.num1;
    a. num 1 = b. num 2;
    b. num 2 = temp;
}
int main () {
  class1 A ;
  class 2 B ;

  A. accept ();
  B. accept ();

  cout << " Before swapping :" ;
    A. display () ;
    B. display () ;
```

```
SwapNumbers (A, B);
cout << "In After swap: In";
  A.display();
  B. display();

return 0;
}
```

```
enter first numbers: 5
enter second numbers: 6

Before swap:
  class1 value : 5
  class2 value : 6

After swap:

  class1 value : 6
  class2 value : 5
```

Q.4 Create class Result 1 & 2 to store student marks. Read value of marks from both class objects & compute average of 2 results.

→

```
#include <iostream>
using namespace std;
class Result 1 {
Private :
float mark 1;
Public :
void accept () {
cout << " Enter marks for first
        student : ";
cin >> marks 1;
}
friend float Average (Result 1, Result 2);
};
class Result 2 {
Private :
float mark 2;
Public :
void accept () {
cout << " enter marks for
        second students : ";
cin >> mark 2;
}
freind float Average (Result 1, Result
                      2)
};

float Average (Result1 a, Result2 b).
{
  float average = (a.mark1 + b.mark
                   2.0, 2).
  return 0;
  return average;
}
int main () {
  Result 1 r1;
  Result 2 r2;
  float avg;
```

```
r1.accept();
r2.accept();

avg = Average (r1, r2);
cout << "Average of two Result
        is : " << endl;

return 0;
}
```

O/P: enter marks for first student: 70
enter marks for ~~student~~
second student: 60
Average of two results: 65

Q.5 WAP to find greatest number among 2 numbers from diff classes using friend functions.

→
```
#include <stdio.h>
using namespace std;

class Number2;
class Number1 {
private:
  int num1;
Public:
void accept() {
  cout << "enter number: ";
  cin >> num1;
}
  friend void findgreatest (Number1,
                            Number2);
};

class Number2 {
private:
  int num2;
Public:
void accept() {
  cout << "enter numbers: ";
  cin >> num;
}
  friend void findgreatest (Number1,
                            Number2);
};
void findgreatest (Number1 a, Number2 b)
{
  if (a.num1 > b.num2) {
    cout << "greatest number
            is:" << a.number1;
    endl;
  }
  else if (b.num2 > a.num1)
  {
    cout << "greatest number is
            : " << b.num2
            << endl;
  } else {
    cout << "Both are equal"
         << endl;
  }
}
```

```
int main () {
  Number 1 obj 1;
  Number 2 obj 2;
    obj 1.accept();
    obj 2.accept();
  find greatest (obj1, obj 2);
  return 0;
}
```

O/P Enter number = 8
enter number = 7
greatest number is : 8

q.6 Create 2 class ~~oo~~, class A and class B each with a private integer. write a friend function sum() that can access private data from both classes and return the sum

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

class classB;
class classA {
  private:
    int valueA
public:
  void input () {
    cout << " enter value for class A: ";
    cin >> valueA;
  }
  friend int sum (classA a, classB b);
};
class classB {
  private:
    int valueB;
Public:
  void input () {
    cout << " enter value for classB: ";
    cin >> valueB;
  }
  friend int sum (classA a, classB, b);
};
  int sum (classA a, classB, b) {
    return a.valueA + b.valueB;
}
int main () {
  Class A objA;
  Class B objB;
  objA.input();
  objB.input();
  cout << "Sum: " << sum (objA, objB)
       << endl;
  return 0;
}
```

O/P :- Enter value for class A : 3
enter value for class B : 5
Sum : 8.

Q-7 write a program with a class Numb. that contain a private integer. Use a friend function swapNumbers (Number &, Number &) to swap the private values of 2 numbers objects.

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

class Number {
private:
    int value;
Public:
  void accept() {
      cin >> value;
  }
  void display() {
      cout << value << endl;
  }
  freind void swapNumbers (Number n1, Number n2);

};
void swapNumbers (Number n1, Number n2)
{
    int temp = n1.value;
     n1.value = n2.value;
      n2.value = temp;
int main() {
   Number num1, num2;
   cout << "enter value for num1:";
    num1.accept();
   cout << "enter value for num2:";
    num2.accept();
   cout << "Before swapping:" << endl;
    cout << "num1 = ";
    num1.display();
    cout << "num2 = ";
     num2.display();
   swapnumbers (num1, num2);
   cout << "After swapping:" << endl;
   cout << "num1 = ";
    num1.display();
    cout << "num2 = ";
    num2.display();

     return 0;
}
```

O/P:
```
Enter value for num1: 5
enter value for num2: 3
Before swapping:
    num1 = 5
    num2 = 3
After swapping:
    num1 = 5
    num2 = 3
```

3. Define 2 classes Book and cube, each having a private volume. Write a friend function find Greater (Box, cube), that determines which object has a larger volume.

```
#include <iostream>
using namespace std;
class cube;
class box{
  private:
    int volume;
Public:
  void accept()
{
  cout << "enter the volume of
        box:";
  cin >> volume;
}
friend void findgreater (box, cube);
};
class cube {
private:
    int volume;
Public:
  void accept()
{
  cout << "enter the volume
      of box:";
  cin >> volume;
}
friend void findgreater (box, cube);
};
class cube {
private:
    int volume;
Public:
  void accept()
{
  cout << "enter the volume of
        cube:";
  cin >> volume;
}
friend void findgreater (box, cube);
};
void findgreater (box b, cube c)
{
    if (b.volume >= c.volume)
      cout << "volume of box
          is greater";
    else if (b.volume == c.volume)
      cout << " both volume are
          equal";
    else
      cout << " volume of cube is
          greater ";
}
int main() {
```

```
box h;
cube c;
h.accept();
c.accept();
findgreater(h,g);
return 0;
}
```

O/P:- Enter the volume of box: 67
enter the volume of cube: 45
volume of box is greater

4. Create a class complex with real and imaginary parts as private members. use a freind function to add 2 complex numbers and return the result as a new complex object.

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

class complex {
private:
    float real;
    float imag;
public:
    void accept() {
        cout << "enter real part:";
        cin >> real;
        cout << "enter Imaginary part:";
        cin >> imag;
    }
    void display() {
        cout << real << "+" << imag << "i" << endl;
    }
    friend complex addcomplex(complex c1, complex c2);
};
complex addcomplex(complex c1, complex c2) {
    complex temp;
    temp.real = c1.real + c2.real;
    temp.imag = c1.imag + c2.imag;
    return temp;
}
int main() {
    complex num1, num2, sum;

    cout << "enter first complex number:" << end;
    num1.accept();

    cout << "enter second complex number:" << endl;
    num2.accept();
    sum = addcomplex(num1, num2);

    cout << "sum of complex number";
    sum.display();
    return 0;
}
```

Q

O/P: enter first complex numbers:
enter real part: 7
enter imaginary part: 5
enter second complex numbers:
enter real part: 3
enter imaginary part: 9
Sum of complex numbers: 10+14i

O/

5. Create a class student with 3 private data members name and 3 subjects marks. write a friend function calculate average (stu) that calculates and displays average marks.

→
```
#include <iostream>
using namespace std;
class student {
private:
    string name;
    float mark1, mark2, mark3;
Public:
    void accept () {
    cout << "enter student name:";
    cin >> name;
    cout << "enter marks in three
        subjects:";
    cin >> mark1 >> mark2 >> mar
            3;
}
friend void calculateAverage(student);
};
void calculatorAverage(students)
{
    float avg = (s.mark1 + s.mark2
                + s.mark3)/3;
cout << "student Name:" << s.name
<< endl;
cout << "Average marks:" << avg <<
    endl;
}
int main () {
    student stu;
    stu.accept();
    CalculateAverage(stu);
    return 0;
}
```

O/P :-
enter student name: Sanika
enter marks in three subjects:
56
89
45
student name: sanika
Average marks: 63.33

6. Create three classes : Alpha, Beta & Gamma, each with a private data member. Write a single friend function that can access all three and print their sum

```
#include <iostream>
using namespace std;
class Beta;
class gamma;

class alpha {
private:
    int a;
public:
    void accept() {
    cout << " enter value for
    Alpha: ";
    cin >> a;
    }
    friend void sumvalus (alpha,
                          Beta,
                          Gramma);
};
class Beta {
    private:
        int b;
    Public:
    void accept() {
    cout << " enter value for
                beta: ";
    cin >> b;
}
```

```
friend void sumvalues (Alpha, Beta,
                       Gamma);
};
class Gamma {
private:
    int c;
public:
    void accept() {
        cout << " enter value for
            gamma: ";
        cin >> c;
    }
    friend void swapvalues(Alpha,
            beta, Gamma);
};
void Sumvalus (Alpha x, Beta y,
               gamma);
};
void sumvalues(Alpha x, Beta y,
               gamma z) {
    int sum = x.a = y.b = z.c;
    cout << " sum of all valus :" <<
        sum << endl;
}

int main() {
    Alpha obj A;
    Beta obj B;
    Gamma obj C;

    ObjA.accept();
    obj B.accept();
    Objc.accept();
```

sumvalues(objA, objB, objC);
return 0;
}

O/P :-
enter value for alpha: 7
enter value for Beta: 4
enter value for gamma: 0
Sum of all values = 11

7. Create a class point with private members x & y. write a friend function that calculates and returns the distance between two point objects.

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

class point {
private:
    float x;
    float y;
public:
    void accept() {
    cout << "enter coordinates (
        x y) :";
    cin >> x >> y;
}
friend float calcDistance(point,
    point);
};

float calcdistance(point p1, point
    p2)
{
    float dx = p2.x - p1.x;
    float dy = p2.y - p1.y;
    return sqrt(dx*dx + dy*dy);
}
int main() {
    Point A, B;
    A.accept();
    B.accept();
    float distance = calcdistance
    (A, B);
    cout << "The distance between
        the two point is :"
        << distance <<
        endl;
    return 0;
}
```

8. O/P :-
enter coordinates (x,y) : 6 7
enter coordinates (x,y) : 6 9
The distance between the 2 point is : 2.8281.

Q 8: Create 2 classes: BankAccount and Audit. BankAccount holds private balance information. Write a friend function in audit that accesses and prints balance information for auditing

→ #include<stdio.h>
using namespace std;

class Audit;

class BankAccount {
Private:
double balance;
Public:
void getbalancefromusers(){
Cout << "enter account balance: ";
cin >> balance;
}
friend void AuditReport(BankAccount account, audit);
};
class audit{
public:
friend void auditreport(Bankaccount, audit);
};

void auditReport(BankAccount, Audit)
{
Cout << "audit Report": Account balance is " <<
account balance <<endl;
}
int main() {
Bank account acc;
audit auditor;
acc.getbalancefromusers();
auditReport(acc, auditor);
}
return 0;
}

Output:-
Enter account balance: ₹677
audit Report: ~~Audit~~ account balance is ₹677.

12/8/25

Exp 3

Q.3 WAP to demonstrate use of nested class

```
#include <iostream>
using namespace std;
class student {
public:
class marks {
public:
int m1, m2, m3;
void getinput() {
cout << "enter marks for 3 subjects";
cin >> m1 >> m2 >> m3;
}
float getpercentage() {
int total = m1 + m2 + m3;
}
float getpercentage() {
int total = m1 + m2 + m3
return total / 3.0;
}
};
};
int main() {
student :: marks marks;
marks.getinput();
float percentage = marks.
getpercentage();
cout << percentage = " <<
percentage << "%";
return 0;
}
```

O/P enter marks for 3 subjects: 95
98
98
percentage : 97.0%

1/2/8/25

Experiment 5

Q.1 write a program to find sum of numbers between 1 to n using a constructor where value of n will be passed to constructor.

→ DEFAULT :-

```
#include <iostream>
using namespace std;
Class num
{
   int n ;
   int sum;
Public :
num()
{
  n = 5;
 sum = 0;
 for (i = 0; i <= n ; i++)
{
    sum + = n i;
  }
}
void display()
{
  cout << "sum is : " << sum;
}
};
int main()
{
   num n1;
  n1. display();
  return 0;
}
```

PARAMETERIZED :-

```
#include <iostream.h>
using namespace std;
class num
{
    int n;
    int sum ;
Public :
num (int no)
{
     n = no;
     for (int i = 0; i <= n ; i++)
     {
          sum + = i ;
     }
}
void display()
{
    cout << " sum is : " << sum
}
};
int main ()
{
    num n1(5);
    n1 . display ();
    return 0;
}
```

Q Q. output :-

Sum is : 15

copy :-

```
#include <iostream>
using namespace std;
class sum
{
    int n;
    int s=0;
    Pubic:
    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;
}
```

Q.2 WAP to declare a class 'student' having dm name & percentage. Write a cons. to initialize these dm. Accept and display for one student.

→ DEFAULT :-

```
#include <iostream>
using namespace std;
class student
{
        string name;
        float per;
Public:
student()
{
        name = "sanika";
        per = 80;
}
void display()
{
        cout <<"name :" << name <<endl;
        cout << "percentage :" << per;
}
};
int main()
{
        Student S1;
        S1.display();
        return 0;
}
```

PARAMETERIZED:-

```
#include <iostream>
using namespace std;
class student
{
    string name;
    float per;
public:
    student (string n, float p)
    {
        name = n;
        per = p;
    }
void display ()
{
    cout << "name:" << name << endl;
    cout << "percentage:" << per;
}
};
int main()
{
    student S1("sanika", 90);
    S1.display();
    return 0;
}
```

Output :-

name : sanika
percentage : 90

COPY :-

```
#include <iostream>
using namespace std;
class student {
private:
    string name;
    float percentage;
Public:
    student (string n, float p)
    {
        name = n;
        percentage = p;
    }
    student (const student &s) {
        name = s.name;
        percentage = s.percentage;
    }
    void display() {
        cout << "name:" << name << endl;
        cout << "percentage:" << percentage
             << "%." << endl;
    }
};
int main()
{
    student S1("Hrishikesh", 92.5);
    student S2 = S1;
    S2.display();
    return 0;
}
```

O/P :-

enter student name : abc
enter percentage : 98

student data
name : abc
Percentage : 98%

3. Define a class 'college' members variables as roll no, name, course WAP using constructor with default value as "computer engeneering" for course. Accept and display that data.

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

class college {
        int roll_no;
        string name;
        string course;
Public:
    college (int r, string n, string
c = " computer science ")
{
        roll no = r;
        name = n;
        course = c;
}
void display ()
{
    cout << " roll no : " << roll_no <<
        endl;
    cout << " Name : " << name << "
            endl;
    cout << " course : " << course
        << endl;
}
};

int main()
{
    int r1, r2;
    string n1, n2;

cout << " enter roll no and name
        for first student : ";
cin >> r1 >> r2;
college c1(r1, n1);
college c2(r2, n2);

cout << " Details of students : ";
s1. display();
s2. display();
return 0;
```

5

O/P :-

enter roll no. and name for first students : 54 abc for

enter rollno and name for second student : 22 xyz

details of students.

Roll no : 54
name : abc
course : computer engineering

Roll no : 22
name : xyz
course : computer engineering

4. WAP to demonstrate constructors overloading

```
#include <iostream>
using namespace std;
class number {
    int x, y;
Public:
    number (int a)
    {
        x = a;
        y = 0;
    }
    number (int a, int b)
    {
        x = a;
        y = b;
    }
    void display()
    {
        cout << "x = " << x << "y = " << y
        << endl;
    }
};
int main()
{
    number n1(5);
    number n2(10, 20);
    n1. display();
    n2. display();
    return 0;
}
```

O/P :- x = 5 y = 0
x = 10 y = 20

16/9/25

Experiment 6

Q Q.1 Create a base class called person with attributes name and age. Derive a class student from person that adds an attribute rollNumbers. Write functions to display all detailed of student

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

class person
{
    protected:
     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
Public : {
        private:
 int roll no;
Public:
    void acceptstudent details()
{
    acceptpersonaldetails();
    cout<<" Enter Roll no:";
    cin >> roll no;
}
    void displaystudentdetails()
{
        displaypersondetails();
 cout <<"Roll no:"<< Roll no<<endl;
};
return 0;
}
int main()
{
  Student S;
  S. acceptstudentdetails();

 cout<<"In Student Details : \n";
 S. displaystudentdetails();

  return 0;
}
```

Output:-

Name: Sanika
Age: 17
Roll no: 1

Q.2 create two base classes academic and sports. Academic class contains marks of student. sports class contain sport score. create a derived class Result that inherits from both academic and sports. write 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 getscore()
{
    cout << "Enter sports score:";
    cin >> score;
}};
class result: public academic public
  sports
{
void display()
{
    int total = marks + score;
    cout << "Academic marks:" << marks;
    cout << "\n Sports score:" << score;
    cout << "\n total score:" << total <<
    endl;
}};
int main()
{
    Result r;
    r.getMarks();
    r.getscore();
    r.display();
    return 0;
}
```

output :-

```
Marks: 90
Sports score: 95
Total score: 185
```

(c) create a class vehicle with attributes like brand and model. Derive a class from car from vechicle which adds an attribute type. Further derive a class, Electric car for car which adds battery capacity write functions 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 Electriccar: 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 (kmh):";
 cin>> battery capacity;
}
void display()
{
 cout<<"Electric car details:";
 cout<<"Brand:"<< brand<<endl;
 cout<<"model:"<<model<<endl;
 cout<<"Type:"<<type<<endl;
 cout<<"Battery capacity:"<< battery
  capacity <<"kmh<<endl;
};
int main()
{
    electric car e;
      e.input;
      e.display;
      return 0;
}.
```

(d)
```
#include <iostream>
#include <string>
using namespace std;

class employee {
  Public:
    int empid;
    string name;
void display()
{
cout<<" employee ID: "<<empid<<endl;
cout<<" Name "<<name<<endl;
}};
class manager : public employee
{
    string department;
    void display()
{
    employee::display();
cout<<" Department: "<<department
    <<endl;
}};
class developer : public employee {
public:
 string programminglanguages;
  void display()
{
employee::display();
 cout<<" programming language:"<<
     programming language<<endl;
}};
```

```
int main()
{
  manager m;
  m.empid = '101'
  m.name = 'ABC';
  m.department = "marketing";
   m.display();
developer d;
   d.empid = 102;
   d.name = 'MANU';
   d.programminglanguage = 'C++';
   d.display();
   return 0;
  }
```

(e) ~~combine~~

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

class person{
Public:
    string name;
     int age;
void display person()
{
    cout<<"Name:"<<name<<
            endl;
    cout<<"Age:"<<age<<endl;
}};
    class student : public person
{
```

```
{
    Public:
    int studentid;
    void displaystudent()
    {
        display person();
    cout << "student ID :" << studentid <<
                endl;

    }};
    class sports{
        public:
        string sportname;
        int sportscore;

    void displaysports()
    {
        cout << "sport : " << sportname <<
                endl.
        cout << " sportscore : " << sportscore
    << endl;
    }};
    class Result : public student, public
        sport {
            public:
            float academic score;
            void displayResult()
            {
                displaystudent();
                displaysport();
    cout << "acedic score : " << academic
                score << endl;
    }};
```

```
int main()
{
    Result r;
    r.name = "sanika";
    r.age = 17;
    r.studentId = 177;
    r.sportname = "Football";
    r.sportscore = 98;
    r.academicscore = 98.5;
    r.displayresult();
    return 0;
}
```

(f)

```
#include<iostream>
using namespace std;
class college : student
{
    protected:
        int student_id;
        int college_code;
    Public:
        void dinputstudent()
        {
            cout << "enter student id :",
            cin >> student_id;
            cout << "enter college code:";
            cin >> college_code;
        }
        void displaystudent()
        {
            cout << "student ID" << student
                    << endl;
```

```
        cout<<" college code :" << college_
            << endl;
    }};
class Test : virtual public college_
             student
{
  protected:
Public:  float percentage;
    void inputTest()
    {
      cout<< "enter test percentage:";
      cin >> percentage;
    }
  voiddisplayTest()
{
  cout << "Test percentage :" << percentage
       << "%" << endl;
}};
class sports : virtual public college_stude
protected:
     char grade;
Public:
  void inputsports()
{
  cout<<" Enter sports Grade :";
   cin >> grade;
}
  void displaysports()
{
  cout <<" sports grade :" << grade
     << endl;
```

```
}};
class Result : pubuc test , public sport
{
 public:
    void inputResult()
    {
      inputstudent();
       input test();
       input sports();
    }
  void displayResults()
{
     displaystudents;
     displaytest();
     displaysports();
}};
int main()
{
     Result r;
     r.inputResult();
     r.displayResult();
     return 0;
}
```

2
17/10

EXP 7

(a)

→
```
# include <iostream>
using namespace std;
class area
{
  public:
    int calculate (int l, int b)
{
  return l*b;
}
  int calculate (int side)
{
  return side * side;
}};
 int main()
{
  area a;
  cout<<" Area of laboratory (Rectangle 10x5):
      << a.calculate (10,5) << endl;
cout<<"area of classroom (Square 6x6):
      << a.calculate (6) << endl;
  return 0;
}
```

b)

→
```
# include <iostream>
using namespace std;
class sum
{
  Public:
  float add (float a, float b, float c,
float d, float e)
{
  return a+b+c+d+e;
}
int add (int a1, int a2, int a3, int a4
      , int a5, int a6, int a7, int
        a8, int a9, int a10)
{
  return
  a1+a2+a3+a4+a5+a6+a7+a8+a9+a10;
}};
int main()
{
  Sum S;
cout<<"Sum of 5 float :"<<S.add(1.1f,
    2.2f, 3.3f, 4.4f, 5.5f)<<endl;
cout<<" Sum of 10 integers :"<<
    S.add (1,2,3,4,5,6,7,8,9,10)
  <<endl;
  return 0;
}
```

(c)

```
# include <iostream>
using namespace std;
Class Number
{
   int x;
  Public:
    Number (int a) { x = a; }
  void display()
{
```

```
    cout << "value:" << x << endl;
}
void operator-()
{
    x = -x;
}};
int main()
{
    Number n(10);
    n.display();
    -n;
    n.display();
    return 0;
}
```

(d)
```
#include <iostream>
using namespace std;
class Number
{
    int x;
Public:
    Number(int a)
{
    x=a;
}
void display()
{
    cout<<"value :"<<x<<endl;
}
void operator++()
{
    ++x;
}};
int main()
{
    number n(5);
    n.display();
    ++n;
    n.display();
    n++;
    n.display();
    return 0;
}
```

17/10

EXP 8

(a)
```
# include <iostream>
# include <string>
using namespace std;
class addstring
{
    string str
Public:
    Addstring (string s)
    {
        str = s;
    }
    Addstring operator+ (addstring obj)
    {
        return Addstring (str + obj.str);
    }
    void display()
    {
        cout << " str << end!;
    }
};
int main()
{
    Addstring S1 ("xyz");
    Addstring S2 ("pqr");
    Addstring S3 = S1 + S2;
cout << "concatenated strings:";
    S3. display();
    return 0;
}
```

(b)
```
# include <iostream>
using namespace std;
class Ilogin
{
    protected:
      String name, passwrated;
    Public:
      virtual void accept()
    {
      cout << "enter name:";
      cin >> name;
      cout << " Enter password: ";
      cin >> password;
    }
      virtual void display()
    {
        cout << " Name: " << name << endl;
        cout << "password: " << password << endl;
    };
    class Emaillogin : public Ilogin
    {
        string email;
    Public:
        void accept()
        override.
    {
      cout << " Email login details:";
      Cout << "Name: " << name << endl;
      Cout << " Password: " << passwrd << endl;
    };
    class membershiplogin : public Ilogin
    {
        int memberid;
```

```
Public:
    void accept()
override
{
    cout<<"membership login:";
    Flogin::accept();
    cout<<"Enter member ID:";
    cin>>memberID;
}
    void display()
override
{
    cout<<"membership login Details:";
    cout<<"Name:"<<name<<endl;
    cout<<"password:"<<password
        <<endl;
    cout<<"member ID:"<<memberID
        <<endl;
}};
int main()
{
    Flogin *login;
    emaillogin e;
    membershiplogin m;

    login=&e;
    login->accept();
    login->display();
    login=&m;
    login->accept();
    Login->display();
    return 0;
}
```

17/10

Exp9

Q.1

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

int main()
{
  fstream new_file;
new_file.open("new_file.txt",ios::out);

if(!new_file)
{
  cout<<"File creation failed"<<endl;
}
else
{
 cout<<"New file created"<<endl;
  new_file.close();
}
 return 0;
}
```

Q.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 file1.txt for
        reading" <<endl;
    return 1;
}
if (!file2)
{
 cout <<" error opening file2.txt for
        writing " <<endl;
    return 1;
}
char ch;
while (file1_read.get(ch))
{
    file2.put(ch);
}
cout << " content copied succesfully from
    file1.txt to file2.txt" << endl;
file1_read.close();
file2.close();
```

```
return 0;
}
```

Q.3: count numbers of digits & spaces

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  ifstream inputfile("Inputtxt");
  if (!inputfile)
{
    cout<< "cannot open input.txt"<<
        endl;
      return 1;
}
char ch;
in spacecount = 0;
digitcount =0;
 while (inputfile.get(ch))
{
    if (ch == ' ') or
        space count++
else if (isdigit(ch))
        digitcount++;
}
   inputfile.close();

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

}
return 0;

Experiment 10

Q.1
→
```
#include <iostream>
using namespace std;
template <class T>
T sumarray (T arr[], int n)
{
    T sum = 0;
    for (int i=0; i<n; i++)
        sum += arr[i];
    return sum;
}
int main()
{
    int int_arr[5] = {1, 2, 3, 4, 5};
    float flo_arr[5] = {1.1, 2.2, 3.3, 4.4, 5.5};
    double dec_arr[5] = {0.5, 1.5, 2.5, 3.5,
                         4.5};
    cout << "sum of integer array: " <<
            sumArray (int_arr, 5) <<
            endl;
    cout << "sum of float array: " << sumA
            rray (flo_arr, 5) << endl;
    cout << "sum of double array: " <<
            sumArray(dec_arr, 5) <<
            endl;
    return 0;
}
```

2)

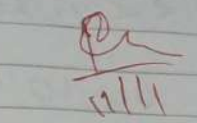
```
#include <iostream>
#include <string>
using namespace std;

template <typename T>
T square (T num)
{
    return num * num;
}
template<>
string square <string> (string str)
{
    return str + str;
}
int main()
{
    int num = 5;
    double dec_num = 2.5;
    string str = "Apple";

cout<< "Square of integer: "<< square
(num) <<endl;
cout << "Square of double: "<< square
(dec_num) << endl;
cout << "square of string: " <<
   square (str) <<endl;

return 0;
}
```

3)

```
#include <iostream>
#include <math.h>
using namespace std;

template <class T1, class T2> class calc
{
  Public:
    T1 x;
    T2 y;

  calc (T1 n , T2 m)
{
      x = n;
      y = m;
}

  void sum()
{
    cout<<"sum: " << x+y << endl;
}
  void diff()
{
    cout << " Difference:" <<x-y << endl;
}
  void pro()
{
    cout<< " Product: " << x* y << endl;
}
  void quo()
{
    cout<<"Quotient:" << x/y << endl;
```

```
}
void rem()
{
 cout<<"Remainder:"<<x%y<<endl;
}
void power()
{
 cout<<"x raised to y is:"<<pow(x,y)
   <<endl;
}
void min_num()
{
 cout<<"min of numbers:"<<fmin(x,y)<<
    endl;
}
void max_num()
{
 cout<<"max of numbers:"<<fmax(x,y)
    <<endl;
}
void sin_x()
{
  cout<<"sin of x:"<<sin(x)<<
      endl;
}
void cos_y()
{
  cout<<"cos of y:"<<cos(y)<<endl
}
};
int main()
{
```

```
calc <int,int> n(100,200);
n.sum();
n.diff();
n.pro();
n.quo();
n.rem();
n.power();
n.min_num();
n.max_num();
n.sin_x();
n.cos_y();
}
```



Exp 11

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

template <class T> class veetor
{
    veetor <T> V ;
Public:
    void createvector (int n)
{
    cout << " Enter " << n << endl;
    v.resize (n);
    for (int i = 0 ; i < n ; i++)
        cin >> v[i];
}
void modify (int index ] value)
{
    if (index > = 0 && index <<
        v.size() )
    v [index] = value;
    else
        cout << " Invalid ! ";
}
void multiply by scaler (T scaler)
{
    for (int i = 0 ; i < v.size () ; i++)
        v [i] *= scaler;
}
void display( )
{
    cout << " ( " ;
```

```
for (int i=0; i<v.size(); i++)
{
    cout<<v[i];
    if(i!=v.size()-1)
        cout<<" ";
}
cout<<"]";
cout<<"\n";
};
int main()
{
    vector n, index, newvalue,
        scalar;

cout<<"Enter size of vector:";
    cin>>n;
    vec.create vector(n);

cout<<"original vector:
    vec.display();

cout<<"\n Enter index & new
    value to modify:";
cin>>index>>newvalue;
vec.modify(index, newvalue);

cout<<"After modification:";
    vec.display();
    }

cout<<"Enter scaler to multiply:";
    cin>>scalar;
```

```
vec.multiply(scalar);
cout<<"After multiplication:";
    vec.display();
    return 0;
};
```

(b) using iterators

```
#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:"<<endl;
for(vector<int>::iterator
        it = v.begin();
    it != v.end(); ++it)
{
    cout<<"*it<<endl;
};

for(vector<int>::iterator it =
    v.begin();
    it!=v.end(); ++it)
{
        cout<<"*it<<" "<<endl;
```

```
}
return 0;
}
```

Experiment 12

Q.1 Implement stack using STL

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

int main()
{
    stack <int> S;
    int ch, x;

do {
cout<<"\n1. push\n2. pop\n 3. Top\n 4.
        Size\n 5. Exit\n";
cout << "enter choice :" ;
  cin>>ch

switch(ch)
{
 case 1: cout<<"enter value :";
         cin >> x;
      S.push(x);
      break;
case 2:
  if ( !S.empty())
{
        S.pop();
    cout<<" popped";
else
```

```
{
cout << "stack is empty";
}
break;
case 3:
if(!s.empty())
{
cout << "top = " << s.top() << endl;
}
else
{
cout << "stack is empty\n";
}
break;
case 4:
cout << "size : " << s.size() << endl;
break;
case 5:
cout << "exit";
break;
default:
cout << "Invalid";
}}
while (ch != 5);
return 0;
}
```

Q.2

```
#include<iostream>
#include<queue>
using namespace std;
int main()
{
queue<int>q;
int ch,x;
do{
cout << "\n 1.enqueue \n 2.dequeue\n 3.front\n 4.size 5.exit\n"
cout << "enter choice:";
cin >> ch;
switch(ch)
{
case 1:
cout << "enter value:";
cin >> x;
q.push();
break;
case 2:
cout <<
if(!q.empty())
{
q.pop();
cout << "dequeue:";
}
else
{
cout << "Queue empty";
```

```
}
break;
case 3:
 if (!q.empty())
{
 cout<< "front :" << q.front();
}
 else
{
 cout << "queue is empty";
}
break;

case 4:
{
 cout<< "size :" << q.size();
  ~~dec~~ break;
e }
cas 5:
 cout << "exit";
 break;

default: {
 cout<< "Invalid";
}
}
while (ch != 5);

 return 0;
}
```