;

Experiment 1-

Q1) C program to find average of 10 numbers using array

```
#include <stdio.h>
int main ()
{
int a[10] ; i ; sum=0;
float average ;
printf ("Enter 10 numbers");
for (i=0 ; i<10 ; i++)
  {
   printf ("number %d", i+1);
   scanf ("%d", &a[i];
  }
  average = sum / 10 ;
printf ("\n sum = %d", sum );
printf (" average = %f", average );

return 0 ;
}
```

Q2) print pattern

```
*
* *
* * *
* * * *
```

```
#include <stdio.h>
int main ()
{
int i, j ;
for (i=0; i<=4; i++)
```

```
{
for (i=0; i<=i; i++)
{
printf ("*");
}
printf ("\n");
}
return 0;
}
```

Q3) print pattern

```
*
* *
* * *
* * * *
```

```
#include <stdio.h>
int main()
{
int i, j;
for (i=0; i<=4; i++)
{
for (j=0; j<=i; j++)
{
if (i%2==0)
{
printf ("*"); }
else
{
printf("*");
}
}
printf ("\n");
}
```

Q3) Power of the given number

```
#include <stdio.h>
int main()
{
int a, b, c;
printf ("Enter the base:");
scanf ("%d", &a);
printf("Enter the exponent:");
scanf("%d", &b);
c = pow (a, b);
printf ("result %d", c);
return 0;
}
```

Q4) Find the first occurring element from the given array

Array

```
#include <stdio.h>
int main()
{
int arr[7] = {2, 6, 2, 4, 6, 8};
int i, j;
int n = sizeof (arr) / sizeof (arr[0]);
int found = 0;
for (i=0; i<n-1; i++)
{
for (j=i+1; j<n; j++)
{
if (arr[i] == arr[j])
```

```
{
printf("first repeating element is %d\n", a[i]);
found = 1;
break;
}
}
if(found==0)
{
printf("no repeat element found");
}
}
return 0;
}
```

(Q5) Squaring odd positioned element of array

```
#include <stdio.h>
int main()
{
int i, arr[] = {1,2,3,4,5,6}, n=6;
for(i=0; i<n; i++)
{
if(i%2 != 0)
{
arr[i] *= arr[i];
}
for(i=0; i<n; i++)
{
printf("%d\n", arr[i]);
}
return 0;
}
```

(Q6) Greatest & smallest element in an array

```
#include <stdio.h>
int main()
{
int a[10], n, i;
int max, min;

printf("Enter number of elements in the array:");
scanf("%d", &n);
printf("Enter %d elements:\n", n);
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}

max = a[0];
min = a[0];

for(i=1; i<n; i++)
{
if(a[i] > max)
{
max = a[i];
}
if(a[i] < min)
{
min = a[i];
}
}
printf("Greatest element = %d\n", max);
return 0;
}
```

11/8/19

Experiment 2 -

PAGE No.
DATE / /

01) Linear Search

```
#include <stdio.h>
int main ()
{
int a[] = {1,2,3,4,5,6};
int key, i;
int found = 0;
printf ("Enter the number to be found: ");
scanf ("%d", &key);
for (i=0; i<6; i++)
{
if (a[i] == key)
{
printf ("%d element found at %d under ", key, i);
}
}
if (!found)
{
printf ("element %d not found in the list: \n")
}
}
if (!found)
{
printf ("element %d not found in the list: \n"
key);
}
return 0;
}
```

2) Declare class Account.

```
#include <iostream>
using namespace std;
class Account {
    public:
    int account_no;
    float balance;
    void accept () {
        cout << "Enter Account No:";
        cin >> account
        cout << "Enter Balance:";
        cin >> balance;
    }
    void display () {
        cout << "Account No:" << account_no << endl;
        cout << "Balance:" << balance << endl
    }
}
int main () {
        Account a[5];
    for (int i=0; i<5; i++){
        a[i].accept ();
    if (a[i].balance >= 5000) {
        a[i].balance += a[i].balance * 0.10;
    }
    }
    cout << "Account with updated: \n";
    for (int i=0; i<5, i++) {
    }
    return 0;
    }
```

1) Declare a class staff having data members.

```
#include <iostream>
#include <string.h>
using namespace std;
class staff {
    public:
        char name[50], post[50]
    void accept () {
        cout << "Enter name of employee";
        cin >> name;
        cout << "Enter the post!";
        cin >> post;
    }
    void display () {
        cout << "Member having post HOD:" << name;
    }
}
int main () {
    staff a[5]
    int i;
    for (i=0; i<5; i++) {
        a[i].accept ();
    }
    for (i=0; i<5; i++) {
        (strcmp (a[i].post, "HOD") == 0) {
            a[i].display
        }
    }
}
```

13/5

Experiment – 3

PAGE NO.
DATE 8/8/25

1) Declare a class 'book' containing data members.

```
#include <iostream>
using namespace std;
class book {
        string book_title;
        string author_name;
        int price;
        void accept() {
            cout << "Enter title:" << endl;
            cin >> this -> book_title;
            cout << "Enter author name" << endl;
            cin >> this -> author_name;
            cout << "Enter price:" << endl
            cin >> this -> price;
    }
    void display() {
        this -> accept();
        cout << "The book title:" << this -> book_title
        cout << "The author:" << this -> author_name << endl;
        cout << "The price:" << this -> price << endl;
        }
        }
int main() {
        book b;
        book *p;
        p = &b;
        p -> display();
    }
```

2) Declare a Class

```
#include <iostream>
#include <string>
using namespace std;
class student {
public:
    int roll_no;
    float per;
    void accept () {
    cout << "Enter roll no: " << endl;
    cin >> this -> roll_no;
    cout << "Enter percentage : " << endl;
    cin >> this -> per;
    }
}
int main () {
    student s;
    s.display ();
}
```

3) Demonstrate nested class

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

class student {
  public:
      class student marks {
        public:
          int roll_no;
          char name [50]
          int m1;
          int m2;

void accept () {
  cout << "Enter roll no: ";
  cin >> roll_no;
  cout << "Enter name : ";
  cin >> name;
  cout << "Enter marks for m1 :";
  cin >> m1;
  cout << "Enter marks for m2 :";
  cin >> m2;
  }
  void display () {
  cout << "The roll no : " << roll_no <<
  cout << "The name : " << name <<
  cout << "Marks for M1 & M2 : <<
```

13/8

```
  }
};
```

*Experiment 4 –

PAGE No.
DATE / /

1. Swap 2 numbers from same using object as function argument

Ans)
```
#include <iostream>
using namespace std;
class number {
int value;
public:
number (int v=0) {
value = v;
}
void swap (number &other) {
int temp = value;
value = other.value;
other.value = temp;
}
void disp () {
cout << "Value:" << value << endl;
}};
int main () {
number n1(10), n2(20);
cout << "Before Swap:" << endl
n1.disp ();
n2.disp ();
return 0;
}
```

O/P:

Before swap:	After Swap:
Value : 10	Value : 20
Value : 20	Value : 10

2. Swap 2 numbers from same class using friend

→
```
#include <iostream>
using namespace std;
class AB {
int a,b;
public:
void Get() {
cout << "Enter 2 numbers: ";
cin >> a >> b;
}
friend void swap (AB &);
};
void swap (AB &A) {
int temp;
temp = A.a;
A.a = A.b;
A.b = temp;
cout << "In Values after swapping: " << A.a << A.b;
}
int main () {
AB A;
A.Get();
swap(A);
}
```

O/P:

```
Enter 2 numbers: 5
                 4
Value after swapping: 4
                      5
```

3. Friend function swap 2 numbers different class

→
```
#include <iostream>
using namespace std;
class CB;
class CA;
int numA;
public:
CA (int val): numA (val) {}
void disp () {
cout << "Value in Class A: " << numA << endl;
}
friend void swap (CA &, CB &);
};
class CB {
private:
int numB;
CB (int val): numB (val) {}
void disp () {
cout << "Value in class B: " << numB << endl;
}
friend void swap (CA &, CB &);
};
void swap (CA &a, CB &b) {
int temp = a.numA;
a.numA = b.numB;
b.numB = temp;
}
```

```
int main () {
CA objA (10);
CB objB (20);
cout << "Before swapping:" << endl;
objA.disp();
objB.disp();
swap (objA, objB);
cout << "\n After swapping:" << endl;
objA.disp();
objB.disp();
return 0;
}
```

O/P:

Before swapping:
Value in class A: 10
Value in class B: 20

After swapping:
Value in class A: 20
Value in class B: 10

4. Avg of two results

```
#include <iostream>
using namespace std;
class result2;
class result {
int a;
public:
void accept () {
cout << "Enter marks out of 50:";
cin >> a;
}
friend void (a) (result x1, result2 x2);
};
class result2{
int b;
Public:
void accept ();
cout << "Enter marks out of 50:";
cin >> b;
}
friend void cal (result x1, result x2);
};
void cal (result x1, result x2) {
}
int main () {
result x;
result 2 y;
x.accept ();
y.accept ();
fal (x,y);
}
```

O/P: Enter marks out of 50: 45
Enter marks out of 50: 46
Avg: 45.5

5. Greatest among 2 numbers using friend function

```
#include <iostream>
using namespace std;
class B;
class A {
int a;
public:
void acc() {
cout << "Enter value:";
cin >> a;
}
friend void gr (A al, Bal);
};
class B {
int b;
public:
void accept() {
cout << "Enter a value:";
cin >> b;
}
friend void gr (A al, B bl);
};
void gr (A al, B bl) {
if (al.a > bl.b) {
cout << "First value is greater";
}
else {
cout << "second value is greater";
}
```

```
int main() {
A x;
B y;
x.acc();
y.acc();
gr (x, y);
}
```

O/P:

Enter value : 10
Enter value : 100
Second value is greater

12/1/

* Experiment - 5

1. Sum default

```
#include <iostream>
using namespace std;
  class sum {
  int n, s=0;
  public:
    sum()
  {
    n=10;
  }
    void display()
  {
    for (int i=0; i<=n; i++)
      s= s+i;
    cout << "Sum is" << s;
    }
  };
    int main()
    {
      sum s1;
      s1. display();
    }
```

O/P:

Sum is 55

1) sum - parameterized

```
#include <iostream>
using namespace std;
class sum
{
 int no, s=0;
public:
 sum (int n)
 {
  no = n;
 }
 void display ()
 {
 for (int i=0; i<= no; i++)
 s= s+i
 cout << "Sum is" << s;
 }
};
 int main ()
 {
  sum s1 (10);
  s1 display ();
 }
```

O/P:

Sum is 55

2) sum - copy constructors

```
#include <iostream>
using namespace std;
class sum {
 int no  s=0
public:
 sum (int n)
 {
  no = n;
 }
 sum (sum & s1)
 {
  no = s1.no;
 }
 void display ()
 {
 for (int i=0; i<= n; i++)
  s= s+i;
 cout << "Sum is" << s;
 }
};
```

4. WAP to declare class student having data members as percentage. Initialize using constructors

* Default

```
#include <iostream>
using namespace std;
class student
{
string name;
int m1, m2;
int total;
public;
Student ()
{
name = "ABC";
m1 = 72;
m2 = 90;
total = 200;
}
void display ()
{
float perc = (float) (m1+m2) / total * 100;
cout << "student name" << name;
cout << "Perantage" << perc;
}
}
int main()
{
S. display();
}
```

★ Experiment 6-

1. Single Inheritance

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

O/P: Enter name Ashik
Enter Age 17
Enter roll no 66

Name Ashik
Age 17
Roll no 66

2) Multiple Inheritance

```
#include <iostream>
using namespace std;
class academic
{
 protected:
 string name,
 int marks;
};
 class sports
{
 protected:
 int score;
};
class student: protected
academic, protected sports
 {
 int total;
 public:
 void accept ()
 {
```

```
cout << "Enter name";
cin >> name;
cout << "Enter marks";
cin >> marks;
cout << "Enter score";
cin >> score;
total = marks + score;
}
void display()
{
cout << "Name" << name;
cout << "Score" << score;
cout << "Total";
}
};
int main()
{
Student s;
s.accept();
s.display();
return 0;
}
```

3. Multilevel

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

```
void display ()
{
cout << "Model" << model;
cout << "Brand" << brand;
cout << "Type" << (type == 1 ? "Sedan" : "SUV");
cout << "Battery Capacity" << battery capacity
}
};
  int main ()
  {
   electrical E;
   e.accept ();
   e.display ();
   return 0;
  }
```

4. Hierarchal

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

class manager : public employee
   {
    private:
    string dep;
   };
class Developer : public employee
   {
        public:
        string lang;
        string dep;
        void accept ()
   {
   cout << "Enter emp name";
   cin >> name;
   cout << "Enter emp id";
   cin >> id;
   cout << "Enter programming language";
   cin >> lang;
   cout << "Enter department";
   cin >> dep;
   }
```

```
void display()
 {
  cout << "Department" << dep;
 }
};
 int main()
 {
  Developer d;
  d.accept();
  d.display();
  return 0;
 }
```

5. hybrid

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

```
class student : public Teacher
 {
  protected:
   int roll;
   string sname;
 }
  void sdisplay()
 {
  cout << "Student Name"
       << sname;
  cout << "Roll number"
       << roll;
  }
 },
  class marks : public student
  {
   protected:
   int m1, m2, m3;
   public:
   void maccept()
  {
   cout << "Enter marks";
   cin >> m1 >> m2 >> m3;
  }
   void mdisplay()
    {
   cout << "Marks in sub 1" << m1;
    cin >> m3;
  }
  };
```

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

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

Ans

Practical

Experiment 7-

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

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

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

```
x, y, z, b;
cout << "Enter Side of square";
cin >> b.Side;
b.calculateArea(b.Side);
cout << "Area of square" << b.area;
return 0;
}
```

O/P:

Enter length & Breadth

3

2

Area of rectangle 6

Enter side of square

5

Area of square 25

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

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

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

c) WAP to implement the operator (for pre increment & post increment) when used with object so that the numeric data members of the class is incremented

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

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

0
1211

*Experiment 8-

a) Write a program to overload the '+' operator so that two strings can be concatenated

```
#include <iostream>
#include <string.h>
using namespace std;
class concate_string {
  public:
  string str;
  &concate_string operator+ (const concate_string & other)
  concate_string result;
  result.str = str + other.str;
    return result;
  } };
  int main () {
  concate_string s1,s2,s3;
  s1.str = "abc";
  s2.str = "xyz";
  s3 = s1+s2;
  s3.display ();
    return 0;
  }
```

* O/P

abcxyz

```
by
#include <iostream>
#include <string>
using namespace std;
class login {
protected
string name;
string password;
public:
string name;
string password;
public
virtual void accept() {
cout<<"enter name:";
cin>> name;
cout<<"enter password";
cin >> password;
}
virtual void display() {
cout<<"Name:"<<name<<endl;
cout >> password;
}};
* class EmailLogin : public login {
private:
string email_ID;
public:
void accept() {
login::accept();
cout<<"email_ID:";
cin>> email_ID;
}
void display() {
login::display();
cout<<"membership type:"<<membership_type;
cin>> ID;
}};
int main() {
Email emp_mail;
membership emp_ms;
cout<<"enter email details:";
emp_email.accept();
cout<<"enter membership:";
emp_ms.accept();
cout<<"email details:";
emp_ms.display();
return 0;
}
```

* Output

```
Enter email details:
Enter name : Adik
-11- password: Adik
-11- membership type: intermediate
-11- ID: 321
Email details:
Name : Adik
Password : Adik
Email ID : adik123@abc.com
Membership details:
Name : Adik
Password : Adik
Membership type: Intermediate
-11- ID: 321
```

12/1

* Experiment 9-

1)
```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
fstream file1, file2;
char ch;
file1.open("first.txt", ios::in);
file2.open("second.txt", ios::out);
while (file1.get(ch)) {
file2.put(ch);
}
cout << "file copied successfully:" << endl;
file1.close();
file2.close();
return 0;
}
```

* O/P
file copied successfully

2)
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
ifstream file("file.txt");
if (!file) {
cout << "unable to open file";
return;
}
else if (ch==' ') {
space_count++;
}}
```

```
cout << "digits: " << digit_count << endl;
cout << "spaces" << space_count << endl;
file.close();
return 0;
}
```

* O/P:

3:
```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
fstream file;
file.open("file.txt", ios::in);
if (file) {
cout << "unable to open file";
return 1;
}
char ch; prev_type = current type;
int
cout << "count: " << word_count << endl;
file.close();
return 0;
}
```

* Experiment 10 -

a)
```
#include <iostream>
using namespace std;
template <class t> t func (t a[], int n) {
t sum = 0;
int i;
for (i=0; i<n; i++) {
sum = sum + a[i];
}
return sum;
}
int main() {
int a1[3] = {2,0,2};
float a2[3] = {1.2, 1.2, 2.4};
double a3[3] = {10.5, 20.5, 30.5};
cout << "sum of integer array is: " << func(a1,3)
<< endl;
cout << "sum of float array is: " << func(a2,3)
<< endl;
cout << "sum of double array is: " << func(a3,3)
<< endl;
return 0;
}
```

* O/P
```
sum of integer array is: 4
sum of float array is: 4.8
sum of double array is: 61.5
```

b) #include <iostream>
using namespace std;
template <class T> T square (T n) {
T result;
result = n*n;
return result;
}
template <> string square <string> (string ss) {
return (ss+ss);
}
int main () {
int i = 2;
string www ("Advik");
ii = square <int> (i);
cout << i << ": " << ii << endl;
cout << square <string> (www) << endl;
return 0;
}

* O/P:
2:4
Advik Advik

c) #include <iostream>
#include <cmath>
using namespace std;
template <class T>
class calculator {
private:
T num1, num2;
public:
Calculator (T n1, T n2) {
num1 = n1;
num2 = n2;
}
T add () {
return num2 + num2;
}
T sub () {
return num1 - num2;
}
T divide () {
if (num2 == 0) {
cout << "error!" << endl;
return 0;
} return num1 / num2
}
void display numbers () {
cout << "Numbers: " << num1 << num2 << endl;
}};

```
cout << "choose an operation";
cout << "1. Addition \n";
```

Ok
Fail

*Experiment 11-

a) To modify the value of a given element

```
#include <iostream>
using namespace std;
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";
  }
  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 (3, 99);
cout << " after modification: ";
v.display ();
return 0;
}
```

* Output
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 &val : vec) {
  val = val * scalar;
 }
 for (int val : vec) {
  cout << val << " ";
 }
 cout << endl;
 return 0;
}
```

* O/P:
3 6 9 12 15

c) To display the vector in 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[i];
 if (i != vec.size()-1)
  cout << ", ";
 }
 cout << ")" << endl;
 return 0;
}
```

* O/P:
(10, 20, 30, 40, 50)

19/11

* Experiment 12-

8. Write a C++ program 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() << endl;
cout << "stack size: " << s.size() << endl;
if (s.empty()) {
  cout << "stack is 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 element:"<<q.front()<<endl;
cout<<"back element:"<<q.back()<<endl;
q.pop();
else{
cout<<"queue is not empty"<<endl;
}
return 0;
}
```

★ O/P:

front element: 10
back element: 20
after pop operation:
front element: 30
queue size: 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;
}

(people.begin(), people.end(),
[](const person& a, const person& b){
    return compare_Age(a,b); });

cout<<"sorted records by age: \n";
  for(auto& p: people){
}
int search_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 == -1)
    cout<<"person with age"<<search_Age<<"fou
```

else
cout << "person with age " << search_Age <<
"not found";
return 0;
}

* O/P:
Sorted records by Age:
Bob - 25
David - 28
Alice - 30
Charlie - 35
Person with age 28 found: David