C++ lab Assignment

Q1. WAP to find a given number is even or odd.

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

Q2. Write a program to find given number is prime or composite.

```
#include<iostream>
using namespace std;
int main()
{
int n;
cout<< “Enter a number:”;
cin>>n;
int count = 0;
for (int i=1;i<=n;i++)
{
if(n%i ==0)
count++;
}
if (count ==2)
{
cout<< “\n Number is prime”;
}
else
cout<< “\n Number is composite”;
return0;
}
```

Q3. Write a program to print table of a given number upto ‘N’ multiple.

```
#include <iostream>
using namespace std;
int main()
{
   int num, n;
   cout << "Enter a number: ";
   cin >> num;
   cout << "Enter number of multiples: ";
   cin >> n;
   cout << "Table of " << num << " up to " << n << " multiples:" << endl;
   for (int i = 1; i <= n; i++)
   {
      cout << num << " x " << i << " = " << num * i << endl;
   }

   return 0;
}
```

Q4. WAP to find:

i) Greater of two numbers
ii) Greater of three numbers

i)

```
#include <iostream>
using namespace std;
int main()
{
   int num1, num2;
   cout << "Enter first number: ";
   cin >> num1;
   cout << "Enter second number: ";
   cin >> num2;
   if (num1 > num2)
  {
      cout << num1 << " is greater.";
   }
else if (num2 > num1)
  {
      cout << num2 << " is greater.";
   } else
  {
      cout << "Both numbers are equal.";
   }
return 0;
}
```

ii)

```
#include <iostream>
using namespace std;
int main() {
   int a, b, c;
   cout << "Enter first number (a): ";
   cin >> a;
   cout << "Enter second number (b): ";
   cin >> b;
   cout << "Enter third number (c): ";
   cin >> c;
   if (a > b && a > c) {
      cout << a << " is greater.";
   } else if (b > a && b > c) {
      cout << b << " is greater.";
   } else if (c > a && c > b) {
      cout << c << " is greater.";
   } else {
      cout << "At least two numbers are equal.";
   }
   return 0;
}
```

Q5. WAP to find sum of first n natural numbers.

```
#include <iostream>
using namespace std;
int main()
{
   int n, sum = 0;
   cout << “Enter the value of n: “;
   cin >> n;
   for (int I = 1; i<= n; i++)
  {
     sum += i;
   }
   cout << “The sum of the first “ << n << “ natural numbers is: “ << sum;
   return 0;
}
```

Q6. WAP to find factorial of a given number.

```
#include <iostream>
using namespace std;
   int main() {
   int num, fact = 1;
   cout << "Enter a number: ";
   cin >> num;
   if (num < 0)
   {
    cout << "Factorial is not defined for negative numbers.";
    }
    else
     {
      for (int i = 1; i <= num; i++) {
         fact *= i;
      }
      cout << "Factorial of " << num << " is: " << fact;
   }
   return 0;
}
```

Q7. Write a program to find sum of digit of n digit natural number.

```
#include <iostream>
using namespace std;
int main()
{
   int num, sum = 0;
   cout << "Enter a number: ";
   cin >> num;
   while (num != 0)
  {
      sum += num % 10;
      num /= 10;
   }
   cout << "Sum of digits is: " << sum;
   return 0;
}
```

Q8. WAP to find reverse of a number.

```
#include <iostream>
using namespace std;
int main()
{
   int num, reverse = 0;
   cout << "Enter a number: ";
   cin >> num;
   while (num != 0)
   {
      reverse = reverse * 10 + num % 10;
      num /= 10;
   }
   cout << "Reverse of the number is: " << reverse;
   return 0;
}
```

Q9. WAP to determine given number is palindrome or not.

```
#include <iostream>
using namespace std;
int main()
{
   int num, reverse = 0, original;
   cout << "Enter a number: ";
   cin >> num;
   original = num;
   while (num != 0)
   {
      reverse = reverse * 10 + num % 10;
      num /= 10;
   }
   if (original == reverse)
   {
      cout << original << " is a palindrome.";
   } else
   {
      cout << original << " is not a palindrome.";
   }
   return 0;
}
```

Q10. WAP to print fibonacci series upto n terms.

```
#include <iostream>
using namespace std;
int main()
{
   int n, t1 = 0, t2 = 1;
   cout << "Enter the number of terms: ";
   cin >> n;
   cout << "Fibonacci Series: " << t1 << " " << t2 << " ";
   for (int i = 3; i <= n; i++)
  {
      int nextTerm = t1 + t2;
      t1 = t2;
      t2 = nextTerm;
      cout << nextTerm << " ";
   }
   return 0;
}
```

Q11. Write a program to determine given n digit number is armstrong ot not.

```
#include <iostream>
Using namespace std;
int main()
{
  int num, originalNum, remainder, result = 0;
  cout << “Enter a three-digit integer: “;
  cin >> num;
  originalNum = num;
  while (originalNum != 0)
  {
    remainder = originalNum % 10;
    result += remainder * remainder * remainder;
    originalNum /= 10;
  }
  if (result == num)
    cout << num << “ is an Armstrong number.”;
  else
    cout << num << “ is not an Armstrong number.”;
  return 0;
}
```

Q12. WAP to print all even numbers between 100 and 200.

```
#include <iostream>
using namespace std;
int main()
 {
   cout << "Even numbers between 100 and 200: "
   for (int i = 100; i <= 200; i++)
{
        if (i % 2 == 0)
    {
       cout << i << endl;
     }
   }
   return 0;
}
```

Q13. Write a program to print first 50 prime numbers.

```
#include<iostream>
using namespace std;
int main()
{
int i,j;
for ( i=1;i<=50;i++)
{
int count=0;
for(j=1;j<=i; j++)
{
if (i%j==0)
count++;
}
if(count==2)
cout<< “\n”<<i;
}
return0;
```

Q14. WAP to print all 4 digit Armstrong numbers.

```
#include <iostream>
using namespace std;
int main()
 {
   cout << "Four-digit Armstrong numbers: " << endl;
   for (int i = 1000; i <= 9999; i++)
{
      int temp = i;
      int sum = 0;
      while (temp != 0)
    {
        int digit = temp % 10;
        sum += digit * digit * digit * digit;
        temp /= 10;
      }
      if (sum == i)
   {
        cout << i << endl;
      }
  }
   return 0;
}
```

Q15. WAP to print following patterns.

i)

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

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

ii)

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

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

iii)

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

```
#include <iostream>
using namespace std;
int main()
{
for (int i=1;i <=3;i++)
{
  for (int j=3; j>i ;j--)
    {
      cout << "    ";
    }
  for (int k=1 ; k<=2*i-1 ; k++)
   {
     cout << "*";
   }
 cout << "\n";
}
return0;
}
```

iv) 1

2 2

3 3 3

4 4 4 4

```
#include <iostream>
using namespace std;
int main()
{
for (int i = 1; i <= 4; i++)
    {
     for (int j = 1; j >= i; j++)
    {
        cout << i;
     }
     cout << endl;
  }
  return 0;
}
```

v) Pascals Triangle

```
#include<iostream>
Using namespace std;
int main() {
int rows,p;
int num=1;
cout <<" Enter the number of rows for Pascal's Triangle: ";
cin >> rows;
for (int i = 0; i < rows; i++) {
p=num;
for (int j=0; j < rows-i-1; j++) {
cout << " ";
}
while(p!=0)
{
int r=p%10;
cout << r <<" ";
p=p/10;
}
num=num *11;
cout << endl;
}
return 0;
}
```

vi) Flyodd's Triangle

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cout << "Enter the number of rows: ";
   cin >> n;
   int num = 1;
   for (int i = 1; i <= n; i++)
 {
      for (int j = 1; j <= i; j++)
    {
        cout << num << " ";
        num++;
      }
      cout << endl;
   }
   return 0;
}
```

Q16. Using functions write following c++ programs.

i) To print all palindrome numbers from range 500 to 1000

```
#include <iostream>
using namespace std;
void Palindrome()
{
   for (int i = 500; i <= 1000; i++)
   {
      int num = i;
      int rev = 0;
      while (num != 0)
     {
         int digit = num % 10;
         rev = rev * 10 + digit;
         num /= 10;
      }
      if (rev == i)
    {
         cout << i << endl;
      }
   }
}
int main() {
  Palindrome ();
return0;
}
```

ii) To print first 100 odd numbers.

```
#include <iostream>
using namespace std;
void Odd()
{
  int n;
 for (n=1; n<=200;n++)
  {
    if (n%2!=0)
     {
        cout << “\n”<<n;
      }
  }
}
int main()
{
  Odd();
  return0;
}
```

iii) To find binary, octal, hexadecimal equivalent of a given decimal number.

```
#include <iostream>
using namespace std;
void binary(int n)
{
   int org_num = n;
   int factor = 1;
   int bin = 0;
   while (n != 0)
   {
      bin = bin + (n % 2) * factor;
      n = n / 2;
      factor = factor * 10;
   }
   cout << "The binary number for " << org_num << " is " << bin << "\n";
}
void octal(int n)
{
   int org_num = n;
   int factor = 1;
   int oct = 0;
   while (n != 0)
   {
      oct = oct + (n % 8) * factor;
      n = n / 8;
      factor = factor * 10;
```

```
    }
    cout << "The octal number for " << org_num << " is " << oct << "\n";
}
void hexadecimal(int n)
{
    int org_num = n;
    int factor = 1;
    int hexa = 0;
    while (n != 0)
    {
        hexa = hexa + (n % 16) * factor;
        n = n / 16;
        factor = factor * 10;
    }
    cout << "The hexadecimal number for " << org_num << " is " << hexa << "\n";
}
int main()
{
    int num;
    cout << "Enter a number:";
    cin >> num;
    binary(num);
    octal(num);
    hexadecimal(num);
    return 0;
}
```

iv) To find decimal equivalents for given binary, hexadecimal and octal numbers.

```
#include <iostream>
#include <cmath>
using namespace std;
void bin(int n)
{
   int org_num = n;
   int deci = 0;
   int power = 0;
   while (n != 0)
   {
      deci = deci + (n % 10) * pow(2, power);
      n = n / 10;
      power++;
   }
   cout << "The decimal number for binary " << org_num << " is " << deci << "\n";
}
void oct(int n)
{
   int org_num = n;
   int deci = 0;
   int power = 0;
   while (n != 0)
   {
      deci = deci + (n % 10) * pow(8, power);
      n = n / 10;
```

```
        power++;
    }
    cout << "The decimal number for octal " << org_num << " is " << deci << "\n";
}
void hex(int n)
{
    int org_num = n;
    int deci = 0;
    int power = 0;
    while (n != 0)
    {
        deci = deci + (n % 10) * pow(16, power);
        n = n / 10;
        power++;
    }
    cout << "The decimal number for hexadecimal " << org_num << " is " << deci << "\n";
}
int main()
{
    int num, choice = 0;
    cout << "Enter 1 if number is binary, 2 if it is octal and 3 if it is hexadecimal:";
    cin >> choice;
    cout << "Enter a number:";
    cin >> num;
    if (choice == 1)
    {
```

```
        bin(num);
    }
    else if (choice == 2)
    {
        oct(num);
    }
    else if (choice == 3)
    {
        hex(num);
    }
    else
    {
        cout << "Invalid choice.";
    }
    return 0;
}
```

v) To calculate geometric sum upto n terms.

```
#include <iostream>
using namespace std;
double geometricSum(double a, double r, int n) {
   double sum = 0;
   double term = a;
   for (int i = 0; i < n; i++) {
      sum += term;
      term *= r;
   }
   return sum;
}
int main() {
   double a, r;
   int n;
   cout << "Enter the first term (a): ";
   cin >> a;
   cout << "Enter the common ratio (r): ";
   cin >> r;
   cout << "Enter the number of terms (n): ";
   cin >> n;
   double sum = geometricSum(a, r, n);
   cout << "Geometric sum up to " << n << " terms: " << sum << endl;
   return 0;
}
```

Q17. Using recursion write following c++ programs.

i) Print binary number for a decimal number

```
#include <iostream>
using namespace std;
void toBin(int n)
{
  if (n > 1)
  {
    toBin(n / 2);
  }
  cout << n % 2;
}
int main()
{
  int num;
  cout << "Enter the number:";
  cin >> num;
  if (num == 0)
  {
    cout << 0;
  }
  else
    toBin(num);
return0;
}
```

ii) Print octal number for a decimal number.

```
#include <iostream>
using namespace std;
void toOct(int n)
{
   if (n > 1)
   {
      toOct(n / 8);
   }
   cout << n % 8;
}
int main()
{
   int num;
   cout << "Enter the number:";
   cin >> num;
   if (num == 0)
   {
      cout << 0;
   }
   else
      toOct(num);
   return 0;
}
```

iii) Print factorials for a given range.

```
#include <iostream>
using namespace std;
int factorial(int n)
{
  if (n == 0 || n == 1)
  {
    return 1;
  }
  else {
    return n * factorial(n - 1);
  }
}
int main() {
  int upper_limit, lower_limit;
  cout << "For range of factorials, enter lower limit:";
  cin >> lower_limit;
  cout << "Enter upper limit:";
  cin >> upper_limit;
  for (int i = lower_limit; i <= upper_limit; i++)  {
    int fact;
    fact = factorial(i);
    cout << "The factorial of " << i << " is " << fact << "\n";
  }
  return 0;
}
```

iv) Print first n terms of fibonacci series.

```
#include <iostream>
using namespace std;
int fibonacci(int n)
{
  if (n == 0)
  {
    return 0;
  }
  else if (n == 1) {
    return 1;
  }
  else  {
    return fibonacci(n - 1) + fibonacci(n - 2);
  }
}
int main() {
  int n;
  cout << "Enter the number of terms for fibonacci series:";
  cin >> n;
  cout << "Fibonacci series:\n";
  for (int i = 0; i < n; i++) {
    cout << fibonacci(i) << " ";
  }
  return 0;
}
```

Q18. WAP to find average of all the elements of 1D array.

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   int arr[n];
   cout << "Enter the elements: ";
   for (int i = 0; i < n; i++)
  {
      cin >> arr[i];
   }
   int sum = 0;
   for (int i = 0; i < n; i++)
  {
      sum += arr[i];
   }
   double average = (double)sum / n;
   cout << "Average: " << average << endl;
   return 0;
}
```

Q19. WAP to find maximum and minimum value of a 1D numeric array.

```
#include <iostream>
using namespace std;
int main() {
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   int arr[n];
   cout << "Enter the elements: ";
   for (int i = 0; i < n; i++) {
      cin >> arr[i];
   }
   int maxVal = arr[0];
   int minVal = arr[0];
   for (int i = 1; i < n; i++) {
      if (arr[i] > maxVal) {
         maxVal = arr[i];
      }
      if (arr[i] < minVal) {
         minVal = arr[i];
      }
   }
   cout << "Maximum value: " << maxVal << endl;
   cout << "Minimum value: " << minVal << endl;
   return 0;
}
```

Q20. Write a program to find transpose of a 2D matrix.

```
#include <iostream>
using namespace std;
int main()
{
   int rows, cols;
   cout << "Enter number of rows: ";
   cin >> rows;
   cout << "Enter number of columns: ";
   cin >> cols;
   int matrix[rows][cols];
   cout << "Enter matrix elements: " << endl;
   for (int i = 0; i < rows; i++)
   {
      for (int j = 0; j < cols; j++)
      {
         cin >> matrix[i][j];
      }
   }
   cout << "Original Matrix: " << endl;
   for (int i = 0; i < rows; i++)
   {
      for (int j = 0; j < cols; j++)
      {
         cout << matrix[i][j] << " ";
      }
```

```
        cout << endl;
    }
    cout << "Transpose Matrix: " << endl;
    for (int i = 0; i < cols; i++)
     {
       for (int j = 0; j < rows; j++)
      {
          cout << matrix[j][i] << " ";
       }
      cout << endl;
    }
    return 0;
}
```

Q21. WAP to add two matrices.

```
#include <iostream>
using namespace std;
int main() {
   int rows, cols;
   cout << "Enter number of rows: ";
   cin >> rows;
   cout << "Enter number of columns: ";
   cin >> cols;
   int matrix1[rows][cols];
   int matrix2[rows][cols];
   cout << "Enter elements of Matrix 1: " << endl;
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         cin >> matrix1[i][j];
      }
   }
   cout << "Enter elements of Matrix 2: " << endl;
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         cin >> matrix2[i][j];
      }
   }
   cout << "Matrix 1: " << endl;
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
```

```
            cout << matrix1[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Matrix 2: " << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << matrix2[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Sum of Matrix 1 and Matrix 2: " << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << matrix1[i][j] + matrix2[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Q22. WAP to multiply 2D matrices.

```
#include <iostream>
using namespace std;
int main() {
   int rows1, cols1, rows2, cols2;
   cout << "Enter number of rows for Matrix 1: ";
   cin >> rows1;
   cout << "Enter number of columns for Matrix 1: ";
   cin >> cols1;
   cout << "Enter number of rows for Matrix 2: ";
   cin >> rows2;
   cout << "Enter number of columns for Matrix 2: ";
   cin >> cols2;
   if (cols1 != rows2) {
      cout << "Matrix multiplication is not possible." << endl;
      return 0;
   }
   int matrix1[rows1][cols1];
   int matrix2[rows2][cols2];
   int result[rows1][cols2];
   cout << "Enter elements of Matrix 1: " << endl;
   for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols1; j++) {
         cin >> matrix1[i][j];
      }
```

```
}
cout << "Enter elements of Matrix 2: " << endl;
for (int i = 0; i < rows2; i++) {
   for (int j = 0; j < cols2; j++) {
      cin >> matrix2[i][j];
   }
}
for (int i = 0; i < rows1; i++) {
   for (int j = 0; j < cols2; j++) {
      result[i][j] = 0;
      for (int k = 0; k < cols1; k++) {
         result[i][j] += matrix1[i][k] * matrix2[k][j];
      }
   }
}
cout << "Matrix 1: " << endl;
for (int i = 0; i < rows1; i++) {
   for (int j = 0; j < cols1; j++) {
      cout << matrix1[i][j] << " ";
   }
   cout << endl;
}
cout << "Matrix 2: " << endl;
for (int i = 0; i < rows2; i++) {
   for (int j = 0; j < cols2; j++) {
      cout << matrix2[i][j] << " ";
```

```
        }
        cout << endl;
    }
    cout << "Result of Matrix Multiplication: " << endl;
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Q23. WAP to sort an array in ascending order.

```
#include <iostream>
using namespace std;
int main() {
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   int arr[n];
   cout << "Enter the elements: ";
   for (int i = 0; i < n; i++) {
      cin >> arr[i];
   }
   cout << "Original array: ";
   for (int i = 0; i < n; i++) {
      cout << arr[i] << " ";
   }
   cout << endl;
   for (int i = 0; i < n; i++) {
      for (int j = i + 1; j < n; j++) {
         if (arr[i] > arr[j]) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
         }
      }
   }
```

```
    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Q24. Write a program to reverse a given string.

```
#include <iostream>
using namespace std;
int main() {
string str, rev;
cout << “Enter a string : ”;
cin>> str;
for ( int i= str.length()-1; i >=0; i --)
{
 rev+=str[i];
}
cout<< “\n Revsersed string : ”<<rev;
return0;
}
```

Q25. Write a program to count all the vowels in a given string.

```
#include <iostream>
using namespace std;
int main()
{
   string str;
   cout << "Enter a string: ";
   cin >> str;
   int vowelCount = 0;
   for (int i = 0; i < str.length(); i++)
    {
      char ch = str[i];
      if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
         ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U')
       {
         vowelCount++;
        }
   }
   cout << "Number of vowels: " << vowelCount << endl;
   return 0;
}
```

Q26. WAP to check if a given string is palindrome or not.

```
#include<iostream>
using namespace std;
int main()
string st:
cout<<"Enter a string \n";
cin>>st:
int flag=0;
int len=st.size();
for (int i=0;i<len/2;i++)
if(st[i]! = st[len-1-i])
{
flag=1;
}
if(flag==0)
cout<<"Palindrome Word":
else
cout<<" Not Palindrome Word":
return 0;
```

Q27. WAP to check if a given string is anagram or not.

```
#include <iostream>
#include <algorithm>
#include <string>
using namespace std;
int main() {
   string str1, str2;
   cout << "Enter first string: ";
   cin >> str1;
   cout << "Enter second string: ";
   cin >> str2;
   // Convert both strings to lowercase
   transform(str1.begin(), str1.end(), str1.begin(), ::tolower);
   transform(str2.begin(), str2.end(), str2.begin(), ::tolower);
   // Sort both strings
   sort(str1.begin(), str1.end());
   sort(str2.begin(), str2.end());
   // Compare sorted strings
   if (str1 == str2) {
      cout << "The strings are anagrams." << endl;
   } else {
      cout << "The strings are not anagrams." << endl;
   }
   return 0;
}
```

Q28. Define a class called Car with attributes such as make, model, and year. Include member functions to set and get these attributes. Create an object of the Car class and demonstrate the use of its member functions.

```
#include <iostream>
#include <string>
using namespace std;
class Car
{
private:
   string make;
   string model;
   int year;
public:
   // Member function to input data
   void inputData() {
      cout << "Enter make: ";
      cin >> make;
      cout << "Enter model: ";
      cin >> model;
      cout << "Enter year: ";
      cin >> year;
   }
   // Member function to output data
   void outputData() {
      cout << "Make: " << make << endl;
      cout << "Model: " << model << endl;
      cout << "Year: " << year << endl;
```

```
    }
};
int main()
{
    Car myCar;
    cout << "Enter car details:" << endl;
    myCar.inputData();
    cout << "\nCar Details:" << endl;
    myCar.outputData();
    return 0;
}
```

Q29. Define a class called Address with attributes such as street, city, and zipCode. Create a class called Person that has an Address object as a member variable. Demonstrate composition by creating a Person object and accessing its Address attributes.

```
#include <iostream>
#include <string>
using namespace std;
class Address {
public:
   string street;
   string city;
   string zipCode;
   Address(string s, string c, string z)
 {
      street = s;
      city = c;
      zipCode = z;
   }
};
class Person {
public:
   string name;
   Address address;
   Person(string n, string s, string c, string z) {
      name = n;
      address = Address(s, c, z); // Initialize Address object
   }
```

```
    // Function to display Person details
    void displayDetails() {
        cout << "Name: " << name << endl;
        cout << "Address: " << address.street << ", " << address.city << " " << address.zipCode
<< endl;
    }
};
int main() {
    Person person("John Doe", "123 Main St", "Anytown", "12345");
    person.displayDetails();
    return 0;
}
```

Q30. Write a program to display the minimum, maximum, sum, search and average of elements of an array.

```
#include <iostream>
using namespace std;
int main() {
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   int arr[n];
   cout << "Enter the elements: ";
   for (int i = 0; i < n; i++) {
      cin >> arr[i];
   }
   // Find minimum
   int minVal = arr[0];
   for (int i = 1; i < n; i++) {
      if (arr[i] < minVal) {
         minVal = arr[i];
      }
   }
   // Find maximum
   int maxVal = arr[0];
   for (int i = 1; i < n; i++) {
      if (arr[i] > maxVal) {
         maxVal = arr[i];
```

```
    }
}
// Calculate sum
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += arr[i];
}
// Search for an element
int searchVal;
cout << "Enter the value to search: ";
cin >> searchVal;
bool found = false;
for (int i = 0; i < n; i++) {
    if (arr[i] == searchVal) {
        found = true;
        break;
    }
}
if (found) {
    cout << "Value found in the array." << endl;
} else {
    cout << "Value not found in the array." << endl;
}
// Calculate average
double average = (double)sum / n;
```

```
    // Display results
    cout << "Minimum value: " << minVal << endl;
    cout << "Maximum value: " << maxVal << endl;
    cout << "Sum: " << sum << endl;
    cout << "Average: " << average << endl;
    return 0;
}
```

Q31. Define a class student with the following specification

Private members of class student

admno integer

sname 20 character

eng. math, science float

total float

Public member function of class student

ctotal() a function to calculate eng + math + science with float return type.

Takedata() Function to accept values for admno, sname, eng, science

Showdata() Function to display all the data members on the screen

```
#include <iostream>
using namespace std;
class Student {
private:
  int admno;
  char sname[20];
  float eng, math, science;
  float total;
public:
  // Function to calculate total
  float ctotal() {
    total = eng + math + science;
    return total;
  }

  void takeData() {
```

```
        cout << "Enter admission number: ";
        cin >> admno;
        cout << "Enter student name: ";
        cin >> sname;
        cout << "Enter English marks: ";
        cin >> eng;
        cout << "Enter Math marks: ";
        cin >> math;
        cout << "Enter Science marks: ";
        cin >> science;
    }
    void showData() {
        cout << "Admission Number: " << admno << endl;
        cout << "Student Name: " << sname << endl;
        cout << "English Marks: " << eng << endl;
        cout << "Math Marks: " << math << endl;
        cout << "Science Marks: " << science << endl;
        cout << "Total Marks: " << ctotal() << endl;
    }
};
int main() {
    Student student;
    student.takeData();
    student.showData();
    return 0;
}
```

Q32. Define a class in C++ with following description:

Private Members

A data member Flight number of type integer

A data member Destination of type string

A data member Distance of type float

A data member Fuel of type float

A member function CALFUEL() to calculate the value of Fuel as per the following criteria

Distance	Fuel
<=1000	500
more than 1000 and <=2000	1100
more than 2000	2200

Public Members

A function FEEDINFO() to allow user to enter values for Flight Number, Destination, Distance & call function CALFUEL() to calculate the quantity of Fuel.

A function SHOWINFO() to allow user to view the content of all the data members.

```
#include <iostream>
#include <string>
using namespace std;
class Flight {
private:
  int flightNumber;
  string destination;
  float distance;
  float fuel;
```

```
    // Member function to calculate fuel
    void calFuel() {
        if (distance <= 1000) {
            fuel = 500;
        } else if (distance > 1000 && distance <= 2000) {
            fuel = 1100;
        } else {
            fuel = 2200;
        }
    }
public:
    // Function to feed information
    void feedInfo() {
        cout << "Enter flight number: ";
        cin >> flightNumber;
        cout << "Enter destination: ";
        cin.ignore(); // Ignore newline character
        getline(cin, destination);
        cout << "Enter distance: ";
        cin >> distance;
        calFuel(); // Calculate fuel
    }
    // Function to show information
    void showInfo() {
        cout << "Flight Number: " << flightNumber << endl;
        cout << "Destination: " << destination << endl;
```

```
        cout << "Distance: " << distance << endl;
        cout << "Fuel: " << fuel << endl;
    }
};
int main() {
    Flight flight;
    flight.feedInfo();
    flight.showInfo();
    return0;
}
```

Q33. Write a menu driven program to perform following:

a) **Input a matrix**
b) **Display matrix**
c) **Add two matrix**
d) **Multiply two matrix**
e) **Transpose a matrix**

```
#include <iostream>

using namespace std;

const int MAX = 10; // Maximum size of the matrix

void inputMatrix(int mat[MAX][MAX], int rows, int cols) {
   cout << "Enter elements of the matrix " << rows << "x" << cols ;
   for (int i = 0; i < rows; i++)
      for (int j = 0; j < cols; j++)
         cin >> mat[i][j];
}

void displayMatrix(int mat[MAX][MAX], int rows, int cols) {
   cout << "Matrix:\n";
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++)
         cout << mat[i][j] << " ";
      cout << endl;
   }
}

void addMatrix(int mat1[MAX][MAX], int mat2[MAX][MAX], int res[MAX][MAX], int rows, int
cols) {
   for (int i = 0; i < rows; i++)
      for (int j = 0; j < cols; j++)
```

```
        res[i][j] = mat1[i][j] + mat2[i][j];
}
void multiplyMatrix(int mat1[MAX][MAX], int mat2[MAX][MAX], int res[MAX][MAX], int r1, int
c1, int c2) {
   for (int i = 0; i < r1; i++)
      for (int j = 0; j < c2; j++) {
         res[i][j] = 0;
         for (int k = 0; k < c1; k++)
            res[i][j] += mat1[i][k] * mat2[k][j];
      }
}
void transposeMatrix(int mat[MAX][MAX], int res[MAX][MAX], int rows, int cols) {
   for (int i = 0; i < rows; i++)
      for (int j = 0; j < cols; j++)
         res[j][i] = mat[i][j];
}
int main() {
   int mat1[MAX][MAX], mat2[MAX][MAX], res[MAX][MAX];
   int r1, c1, r2, c2;
   int choice;
   do {
      cout << "\nMenu:\n";
      cout << "1. Input a matrix\n";
      cout << "2. Display matrix\n";
      cout << "3. Add two matrices\n";
      cout << "4. Multiply two matrices\n";
```

```
cout << "5. Transpose a matrix\n";
cout << "6. Exit\n";
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {
  case 1:
    cout << "Enter rows and columns of matrix: ";
    cin >> r1 >> c1;
    inputMatrix(mat1, r1, c1);
    break;
  case 2:
    displayMatrix(mat1, r1, c1);
    break;
  case 3:
    cout << "Enter rows and columns of both matrices: ";
    cin >> r1 >> c1;
    inputMatrix(mat1, r1, c1);
    inputMatrix(mat2, r1, c1);
    addMatrix(mat1, mat2, res, r1, c1);
    displayMatrix(res, r1, c1);
    break;
  case 4:
    cout << "Enter rows and columns of first matrix: ";
    cin >> r1 >> c1;
    cout << "Enter rows and columns of second matrix: ";
    cin >> r2 >> c2;
```

```
            if (c1 != r2) {
                cout << "Matrix multiplication not possible!\n";
                break;
            }
            inputMatrix(mat1, r1, c1);
            inputMatrix(mat2, r2, c2);
            multiplyMatrix(mat1, mat2, res, r1, c1, c2);
            displayMatrix(res, r1, c2);
            break;
        case 5:
            cout << "Enter rows and columns of matrix: ";
            cin >> r1 >> c1;
            inputMatrix(mat1, r1, c1);
            transposeMatrix(mat1, res, r1, c1);
            displayMatrix(res, c1, r1);
            break;
        case 6:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice! Try again.\n";
    }
  } while (choice != 6);
  return0;
}
```