WEEK 1

Q1

**1. Define a class EMPLOYEE contains following members.   
 Data members: Employee\_Number, Employee\_Name, Basic, DA, IT, Net\_Sal, Gross\_salary.  
Member functions: To read the data, calculate net salary, gross salary and display both salary.  
Write a C++ program to read the data of N employees and compute Net salary and Gross salary of each employee. (DA= 12% of Basic and Income Tax (IT) = 18% of the gross salary).**

#include <iostream>

#include <string>

using namespace std;

class EMPLOYEE {

private:

int Employee\_Number;

string Employee\_Name;

double Basic;

double DA;

double IT;

double Gross\_salary;

double Net\_Sal;

public:

void readData() {

cout << "Enter Employee Number: ";

cin >> Employee\_Number;

cout << "Enter Employee Name: ";

cin.ignore();

getline(cin, Employee\_Name);

cout << "Enter Basic Salary: ";

cin >> Basic;

}

void calculateSalary() {

DA = 0.12 \* Basic;

Gross\_salary = Basic + DA;

IT = 0.18 \* Gross\_salary;

Net\_Sal = Gross\_salary - IT;

}

void displaySalary() const {

cout << "\nEmployee Number: " << Employee\_Number;

cout << "\nEmployee Name: " << Employee\_Name;

cout << "\nBasic Salary: " << Basic;

cout << "\nDearness Allowance (DA): " << DA;

cout << "\nGross Salary: " << Gross\_salary;

cout << "\nIncome Tax (IT): " << IT;

cout << "\nNet Salary: " << Net\_Sal << endl;

}

};

int main() {

int N;

cout << "Enter the number of employees: ";

cin >> N;

EMPLOYEE employees[N];

for (int i = 0; i < N; ++i) {

cout << "\nEnter details for employee " << (i + 1) << ":\n";

employees[i].readData();

employees[i].calculateSalary();

}

cout << "\nSalary details of all employees:\n";

for (int i = 0; i < N; ++i) {

employees[i].displaySalary();

cout << "-----------------------------------\n";

}

return 0;

}

**INPUT:**

Enter the number of employees: 2

Enter details for employee 1:

Enter Employee Number: 101

Enter Employee Name: Mayank

Enter Basic Salary: 20000

Enter details for employee 2:

Enter Employee Number: 102

Enter Employee Name: Saaket

Enter Basic Salary: 200000

**OUTPUT:**

Salary details of all employees:

Employee Number: 101

Employee Name: Mayank

Basic Salary: 20000

Dearness Allowance (DA): 2400

Gross Salary: 22400

Income Tax (IT): 4032

Net Salary: 18368

-----------------------------------

Employee Number: 102

Employee Name: Saaket

Basic Salary: 200000

Dearness Allowance (DA): 24000

Gross Salary: 224000

Income Tax (IT): 40320

Net Salary: 183680

Q2

**2. Create a flight class that has private data members: flight number (integer), destination(characters), distance (float), fuel (float).  
  
a) Initialize fuel to 13.2 liters  
b) Provide a parameterized function that accepts fuel details  
c) Private Member functions: calculate\_fuel() to calculate the value of Fuel as per the following criteria:  
 Distance (in kilometers) Fuel (in liters)  
 <=1000 500  
 >1000 and <=2000 1100  
 >2000 2200  
  
d) Member functions: information\_entry() to allow user to enter values for  
flight number, destination, distance which calls function calculate\_fuel() to  
calculate the quantity of fuel and display\_info() to allow user to view flight details.**

#include <iostream>

#include <string>

using namespace std;

class Flight {

private:

int flight\_number;

string destination;

float distance;

float fuel;

void calculate\_fuel() {

if (distance <= 1000)

fuel = 500;

else if (distance > 1000 && distance <= 2000)

fuel = 1100;

else

fuel = 2200;

}

public:

Flight() {

fuel = 13.2;

}

void information\_entry() {

cout << "Enter flight number: ";

cin >> flight\_number;

cout << "Enter destination: ";

cin >> destination;

cout << "Enter distance (in kilometers): ";

cin >> distance;

calculate\_fuel();

}

void display\_info() {

cout << "Flight Number: " << flight\_number << endl;

cout << "Destination: " << destination << endl;

cout << "Distance: " << distance << " km" << endl;

cout << "Fuel Required: " << fuel << " liters" << endl;

}

};

int main() {

Flight flight;

flight.information\_entry();

flight.display\_info();

return 0;

}

**INPUT:**

**Enter flight number: 101**

**Enter destination: Lucknow**

**Enter distance (in kilometers): 2500**

**OUTPUT:**

**Flight Number: 101**

**Destination: Lucknow**

**Distance: 2500 km**

**Fuel Required: 2200 liters**

**WEEK 2**

**Q1**

Mapping of 2-D arrays to 1-D arrays: Map the following 2-D arrays (matrices) to 1-D arrays (lists). a) Upper triangular matrix b) Lower triangular matrix c) Diagonal matrix d) Tri-diagonal matrix e) Row-major f) Column-major Display the element at any specified position (row, column).

#include <iostream>

using namespace std;

#include <cstdlib>

void upperTraingularMatrix(int n[][100], int len)

{

int ans[((len\*(len+1))/2)];

int counter = 0;

for(int i = 0; i < len; i++)

{

for(int j = i; j < len; j++)

{

ans[counter++] = n[i][j];

}

}

cout << "\nUpper Triangular Matrix: ";

for(int i = 0; i < ((len\*(len+1))/2); i++)

{

cout << ans[i] << " ";

}

}

void lowerTriangularMatrix(int n[][100], int len)

{

int ans[((len\*(len+1))/2)];

int counter = 0;

for(int i = 0; i < len; i++)

{

for(int j = 0; j < i + 1; j++)

{

ans[counter++] = n[i][j];

}

}

cout << "\nLower Triangular Matrix: ";

for(int i = 0; i < ((len\*(len+1))/2); i++)

{

cout << ans[i] << " ";

}

}

void diagonalMatrix(int n[][100], int len)

{

int ans[len];

for(int i = 0; i < len; i++)

ans[i] = n[i][i];

cout << "\nDiagonal Matrix: ";

for(int i = 0; i < len; i++)

{

cout << ans[i] << " ";

}

}

void triDiagonalMatrix(int n[][100], int len)

{

int ans[3 \* len - 2];

int counter = 0;

for(int i = 0; i < len; i++)

for(int j = i - 1; j < i + 2; j++)

{

if(j < 0)

j = 0;

ans[counter++] = n[i][j];

}

cout << "\nTri-Diagonal Matrix: ";

for(int i = 0; i < 3 \* len - 2; i++)

{

cout << ans[i] << " ";

}

}

void rowMajor(int n[][100], int len)

{

int ans[len \* len];

int counter = 0;

for(int i = 0; i < len; i++)

{

for(int j = 0; j < len; j++)

{

ans[counter++] = n[i][j];

}

}

cout << "\nRow Major Matrix: ";

for(int i = 0; i < len \* len; i++)

{

cout << ans[i] << " ";

}

}

void columnMajor(int n[][100], int len)

{

int ans[len \* len];

int counter = 0;

for(int i = 0; i < len; i++)

{

for(int j = 0; j < len; j++)

{

ans[counter++] = n[j][i];

}

}

cout << "\nRow Major Matrix: ";

for(int i = 0; i < len \* len; i++)

{

cout << ans[i] << " ";

}

}

int main()

{

int n;

cout << "Enter the size of the 2D Array: ";

cin >> n;

int matrix[n][100];

for(int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

matrix[i][j] = 1 + rand() % 10;

for(int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

cout << matrix[i][j] << " ";

cout << endl;

}

upperTraingularMatrix(matrix, n);

lowerTriangularMatrix(matrix, n);

diagonalMatrix(matrix, n);

triDiagonalMatrix(matrix, n);

rowMajor(matrix, n);

columnMajor(matrix, n);

return 0;

}

**Enter the size of the 2D Array: 3**

**4 7 8**

**6 8 6**

**7 3 1O**

**OUTPUT:**

**Upper Triangular Matrix: 4 7 8 4 6 10**

**Lower Triangular Matrix: 4 6 4 7 3 10**

**Diagonal Matrix: 4 4 10**

**Tri-Diagonal Matrix: 4 7 6 4 6 3 10**

**Row Major Matrix: 4 7 8 6 4 6 3 10**

**Row Major Matrix: 4 6 7 7 4 3 8 6 10**

**Q2**

Representation of a Sparse Matrix:- Represent a sparse matrix using 1-D array. Use this 1-D array to reconstruct the original matrix.

#include <iostream>

using namespace std;

const int MAX\_NON\_ZERO = 1000; // Arbitrary large number

int main() {

int k, m;

cout << "Enter the number of Rows:\t";

cin >> k;

cout << "Enter the number of Columns:\t";

cin >> m;

int arr[k][m];

cout << "Enter the entries of the Sparse Matrix:\n";

for (int i = 0; i < k; i++) {

for (int j = 0; j < m; j++) {

cin >> arr[i][j];

}

cout << "\n";

}

cout << "The Sparse Matrix: \n";

for (int i = 0; i < k; i++) {

for (int j = 0; j < m; j++) {

cout << arr[i][j] << "\t";

}

cout << "\n";

}

cout << "\n";

int sparseArray[3 \* MAX\_NON\_ZERO];

int index = 0;

for (int i = 0; i < k; i++) {

for (int j = 0; j < m; j++) {

if (arr[i][j] != 0) {

sparseArray[index++] = i;

sparseArray[index++] = j;

sparseArray[index++] = arr[i][j];

}

}

}

int nonZeroCount = index / 3;

cout << "The 1-D Conversion of the Sparse Matrix is:\n";

cout << "Row\tColumn\tValue\n";

for (int i = 0; i < nonZeroCount; i++) {

cout << sparseArray[3\*i] << "\t" << sparseArray[3\*i + 1] << "\t" << sparseArray[3\*i + 2] << "\n";

}

cout << "\n";

int reconstructedMatrix[k][m] = {0};

for (int i = 0; i < nonZeroCount; i++) {

int row = sparseArray[3\*i];

int col = sparseArray[3\*i + 1];

int value = sparseArray[3\*i + 2];

reconstructedMatrix[row][col] = value;

}

cout << "The Reconstructed Sparse Matrix:\n";

for (int i = 0; i < k; i++) {

for (int j = 0; j < m; j++) {

cout << reconstructedMatrix[i][j] << "\t";

}

cout << "\n";

}

return 0;

}

**INPUT:**

**Enter the number of Rows: 3**

**Enter the number of Columns: 3**

**Enter the entries of the Sparse Matrix:**

**1**

**0**

**0**

**0**

**2**

**0**

**0**

**0**

**3**

**OUTPUT:**

**The Sparse Matrix:**

**1 0 0**

**0 2 0**

**0 0 3**

**The 1-D Conversion of the Sparse Matrix is:**

**Row Column Value**

**0 0 1**

**1 1 2**

**2 2 3**

**The Reconstructed Sparse Matrix:**

**1 0 0**

**0 2 0**

**0 0 3**

**WEEK 3:**

**Q1:**

Representation of a Polynomial:-

Represent a polynomial using 1-D array and perform addition operation on

two polynomials.

#include<iostream>

using namespace std;

int max(int m ,int n) { return m>n ? m:n;}

int\* add(int A[] , int B[] , int m,int n)

{

int size =max(m,n);

int\* sum= new int[size];

for(int i=0; i<m; i++)

sum[i]=A[i];

for(int i=0;i<n;i++)

sum[i]=sum[i]+B[i];

return sum;

}

void printPoly(int poly[] , int n)

{

for(int i=0;i<n;i++){

cout <<poly[i];

if(i!=0)

cout<< "x^"<<i;

if(i!=n-1)

cout<< "+";

}

}

int main()

{

int A[] = { 5,0,10,6};

int B[]= {1,2,4};

int m = sizeof(A)/ sizeof(A[0]);

int n= sizeof(B) / sizeof(B[0]);

cout << "First Polynomial is \n";

printPoly(A,m);

cout << "\nSecond Polynomial is \n";

printPoly(B,n);

int\* sum=add(A,B,m,n);

int size=max(m,n);

cout << "\nsum polynomial is \n";

printPoly(sum,size);

return 0;

}

**INPUT:**

int A[] = { 5,0,10,6};

int B[]= {1,2,4};

**OUTPUT:**

First Polynomial is

5+0x^1+10x^2+6x^3

Second Polynomial is

1+2x^1+4x^2

sum polynomial is

6+2x^1+14x^2+6x^3

**Q2**

2. Write a program to perform following string operations without using string

handling functions:

a) length of the string

b) string concatenation

c) string comparison

d) to insert a sub string

e) to delete a substring

#include <iostream>

using namespace std;

int stringLength(char str[]) {

int length = 0;

while (str[length] != '\0') {

length++;

}

return length;

}

void stringConcat(char dest[], char src[]) {

int destLength = stringLength(dest);

int i = 0;

while (src[i] != '\0') {

dest[destLength + i] = src[i];

i++;

}

dest[destLength + i] = '\0';

}

bool stringCompare(char str1[], char str2[]) {

int i = 0;

while (str1[i] != '\0' && str2[i] != '\0') {

if (str1[i] != str2[i]) {

return false;

}

i++;

}

if (str1[i] == '\0' && str2[i] == '\0') {

return true;

} else {

return false;

}

}

void stringInsert(char str[], char sub[], int pos) {

int strLen = stringLength(str);

int subLen = stringLength(sub);

for (int i = strLen; i >= pos; i--) {

str[i + subLen] = str[i];

}

for (int i = 0; i < subLen; i++) {

str[pos + i] = sub[i];

}

}

void stringDelete(char str[], char sub[]) {

int strLen = stringLength(str);

int subLen = stringLength(sub);

int i, j;

for (i = 0; i <= strLen - subLen; i++) {

for (j = 0; j < subLen; j++) {

if (str[i + j] != sub[j]) {

break;

}

}

if (j == subLen) {

for (int k = i; k <= strLen - subLen; k++) {

str[k] = str[k + subLen];

}

str[strLen - subLen] = '\0';

break;

}

}

}

int main() {

char str1[100] = "Hello";

char str2[100] = "World";

char sub[100] = "Friend";

cout << "Length of str1: " << stringLength(str1) << endl;

stringConcat(str1, str2);

cout << "After concatenation: " << str1 << endl;

cout << "Comparison of str1 and str2: " << (stringCompare(str1, str2) ? "Equal" : "Not Equal") << endl;

stringInsert(str1, sub, 5);

cout << "After inserting substring: " << str1 << endl;

stringDelete(str1, sub);

cout << "After deleting substring: " << str1 << endl;

return 0;

}

**INPUT:**

char str1[100] = "Hello";

char str2[100] = "World";

char sub[100] = "Friend";

**OUPUT:**

**Length of str1: 5**

**After concatenation: HelloWorld**

**Comparison of str1 and str2: Not Equal**

**After inserting substring: HelloFriendWorld**

**After deleting substring: HelloWorld**

**WEEK 4:**

**Q**

1. Solving problems using Recursion:
2. **Factorial of a given number**

#include<iostream>

using namespace std;

int factorial(int n){

if(n<=1)

{

return 1;

}

return n\*factorial(n-1);

}

int main()

{

int a;

cout<<"Enter the Number:"<<endl;

cin>>a;

cout<<"The factorial of the entered number is:"<<factorial(a)<<endl;

return 0;

}

**INPUT:**

**Enter the Number:**

**5**

**OUTPUT:**

**The factorial of the entered number is:120**

1. **GCD of 2 numbers**

#include <iostream>

using namespace std;

int gcd(int m, int n) {

if (n == 0) {

return m;

}

return gcd(n, m % n);

}

int main() {

int a, b;

cout << "Enter the first number for finding GCD: " << endl;

cin >> a;

cout << "Enter the second number for finding GCD: " << endl;

cin >> b;

if (a < 0) a = -a;

if (b < 0) b = -b;

cout << "The GCD of the entered numbers is: " << gcd(a, b) << endl;

return 0;

}

**INPUT:**

**Enter the first number for finding GCD:**

**10**

**Enter the second number for finding GCD:**

**100**

**OUTPUT:**

**The GCD of the entered numbers is: 10**

1. **Fibonacci series upto nth term**

#include <iostream>

using namespace std;

int fib(int x) {

if((x==1)||(x==0)) {

return(x);

}else {

return(fib(x-1)+fib(x-2));

}

}

int main() {

int x , i=0;

cout << "Enter the number of terms of series : ";

cin >> x;

cout << "\nFibonnaci Series : ";

while(i < x) {

cout << " " << fib(i);

i++;

}

return 0;

}

**INPUT:**

**Enter the number of terms of series : 6**

**OUPUT:**

**Fibonnaci Series : 0 1 1 2 3 5**

1. **Tower of Hanoi for n disks**

#include<iostream>

using namespace std;

void TOH(int n,char Sour, char Aux,char Des)

{

if(n==1)

{

cout<<"Move Disk "<<n<<" from "<<Sour<<" to "<<Des<<endl;

return;

}

TOH(n-1,Sour,Des,Aux);

cout<<"Move Disk "<<n<<" from "<<Sour<<" to "<<Des<<endl;

TOH(n-1,Aux,Sour,Des);

}

//main program

int main()

{

int n;

cout<<"Enter no. of disks:";

cin>>n;

//calling the TOH

TOH(n,'A','B','C');

return 0;

}

**INPUT:**

**Enter no. of disks:5**

**OUTPUT:**

**Move Disk 1 from A to C**

**Move Disk 2 from A to B**

**Move Disk 1 from C to B**

**Move Disk 3 from A to C**

**Move Disk 1 from B to A**

**Move Disk 2 from B to C**

**Move Disk 1 from A to C**

**Move Disk 4 from A to B**

**Move Disk 1 from C to B**

**Move Disk 2 from C to A**

**Move Disk 1 from B to A**

**Move Disk 3 from C to B**

**Move Disk 1 from A to C**

**Move Disk 2 from A to B**

**Move Disk 1 from C to B**

**Move Disk 5 from A to C**

**Move Disk 1 from B to A**

**Move Disk 2 from B to C**

**Move Disk 1 from A to C**

**Move Disk 3 from B to A**

**Move Disk 1 from C to B**

**Move Disk 2 from C to A**

**Move Disk 1 from B to A**

**Move Disk 4 from B to C**

**Move Disk 1 from A to C**

**Move Disk 2 from A to B**

**Move Disk 1 from C to B**

**Move Disk 3 from A to C**

**Move Disk 1 from B to A**

**Move Disk 2 from B to C**

**Move Disk 1 from A to C**