**PART 2 Assignment for Internal Assessment**

**Q14)**

**SOLUTION**

#include <iostream>

using namespace std;

float area(float r){

return 3.14 \* r \* r;

}

float circumference(float r){

return 3.14 \* r \* 2;

}

int main()

{

float r;

cout << "Enter radius of circle: ";

cin >> r;

if (r < 0){

cout << "Radius can not be negative" << endl;

exit(1);

}

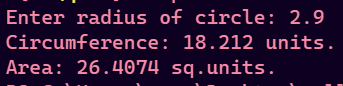
cout << "Circumference: " << circumference(r) << " units." << endl;

cout << "Area: " << area(r) << " sq.units." << endl;

return 0;

}

**OUTPUT**

****

**Q17)**

**SOLUTION**

#include <iostream>

using namespace std;

void print\_arr(int arr[] , int n){

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

{

cout << arr[i] << " ";

}

cout << endl;

}

int main()

{

int n , m , p;

cout << "Enter lengths of array 1 and 2: " << endl;

cin >> n >> m;

if (n <= 0 || m <= 0){

cout << "Array length can not be zero or negative" << endl;

exit(1);

}

int arr1[n];

int arr2[m];

p = n + m;

int arr[p];

cout << "Enter elements of array 1" << endl;

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

{

cin >> arr1[i];

}

cout << "Enter elements of array 2" << endl;

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

{

cin >> arr2[i];

}

// Merging

int i = 0 , j = 0 , k = 0;

while (i < n || j < m){

if (i < n && j < m ){

if (arr1[i] < arr2[j]){

arr[k] = arr1[i];

i++;

}

else{

arr[k] = arr2[j];

j++;

}

}

else if (i < n){

arr[k] = arr1[i];

i++;

}

else{

arr[k] = arr2[j];

j++;

}

k++;

}

cout << "Array 1: ";

print\_arr(arr1 , n);

cout << "Array 2: ";

print\_arr(arr2 , m);

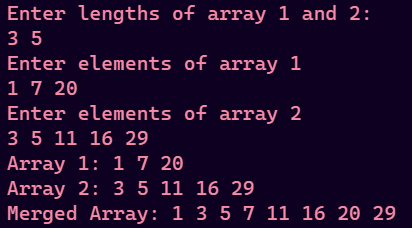
cout << "Merged Array: ";

print\_arr(arr , p);

return 0;

}

**OUTPUT**

****

**Q21)**

**SOLUTION**

#include <iostream>

using namespace std;

class Matrix

{

private:

int n, m;

int arr[100][100];

int transposeArr[100][100];

public:

friend Matrix operator+(Matrix m1, Matrix m2);

friend Matrix operator-(Matrix m1, Matrix m2);

friend Matrix operator\*(Matrix m1, Matrix m2);

friend ostream &operator<<(ostream &output, Matrix &mat);

friend istream &operator>>(istream &input, Matrix &mat);

Matrix() {}

Matrix(int a, int b)

{

n = a;

m = b;

}

void printMatrix()

{

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

{

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

{

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

}

cout << endl;

}

}

void printTranspose()

{

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

{

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

{

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

}

cout << endl;

}

}

void inputMatrix()

{

cout << "Enter elements of matrix" << endl;

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

{

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

{

cin >> arr[i][j];

}

}

}

void transpose()

{

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

{

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

{

transposeArr[j][i] = arr[i][j];

}

}

}

};

ostream &operator<<(ostream &output, Matrix &mat)

{

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

{

for (int j = 0; j < mat.m; j++)

{

output << mat.arr[i][j] << " ";

}

output << endl;

}

return output;

}

istream &operator>>(istream &input, Matrix &mat)

{

cout << "Enter elements of matrix" << endl;

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

{

for (int j = 0; j < mat.m; j++)

{

input >> mat.arr[i][j];

}

}

return input;

}

Matrix operator+(Matrix m1, Matrix m2)

{

Matrix res;

res.m = m1.m;

res.n = m1.n;

for (int i = 0; i < m1.n; i++)

{

for (int j = 0; j < m1.m; j++)

{

res.arr[i][j] = m1.arr[i][j] + m2.arr[i][j];

}

}

return res;

};

Matrix operator-(Matrix m1, Matrix m2)

{

Matrix res;

res.m = m1.m;

res.n = m1.n;

for (int i = 0; i < m1.n; i++)

{

for (int j = 0; j < m1.m; j++)

{

res.arr[i][j] = m1.arr[i][j] - m2.arr[i][j];

}

}

return res;

};

Matrix operator\*(Matrix m1, Matrix m2)

{

Matrix res;

res.n = m1.n;

res.m = m2.m;

int sum;

// n -> rows

// m -> cols

for (int i = 0; i < m1.n; i++)

{

for (int j = 0; j < m2.m; j++)

{

sum = 0;

for (int k = 0; k < m1.m; k++)

{

sum += m1.arr[i][k] \* m2.arr[k][j];

}

res.arr[i][j] = sum;

}

}

return res;

};

int main()

{

Matrix result;

char choice;

int n1, m1, m2, n2;

cout << "Enter dimension of matrix 1 : ";

cin >> n1 >> m1;

cout << "Enter dimension of matrix 2 : ";

cin >> n2 >> m2;

if (n1 <= 0 || m1 <= 0 || m2 <= 0 || n2 <= 0)

{

cout << "Matrix can not have any negative or zero order !!!" << endl;

exit(1);

}

Matrix A(n1, m1), B(n2, m2);

bool isAddOrSubtract = (n1 == n2) && (m1 == m2);

bool isMultiply = m1 == n2;

cout << "For matrix 1" << endl;

cin >> A;

cout << "For matrix 2" << endl;

cin >> B;

// Calculate transposes

A.transpose();

B.transpose();

while (true)

{

cout << "-------MENU-------" << endl;

cout << "1.Print The Matrices" << endl;

cout << "2.Add The Matrices" << endl;

cout << "3.Subtract The Matrices" << endl;

cout << "4.Multiply The Matrices" << endl;

cout << "5.Transpose of matrices" << endl;

cout << "6.Exit" << endl;

cout << "Enter your choice" << endl;

cin >> choice;

if (choice == '6')

{

cout << "Exiting the Program..." << endl;

break;

}

switch (choice)

{

case '1':

cout << "Matrix A" << endl;

cout << A;

cout << "Matrix B" << endl;

cout << B;

break;

case '2':

if (isAddOrSubtract)

{

result = A + B;

cout << "Addition Result" << endl;

cout << result;

}

else

{

cout << "A and B can't be added" << endl;

}

break;

case '3':

if (isAddOrSubtract)

{

result = A - B;

cout << "Subtraction Result" << endl;

cout << result;

}

else

{

cout << "A and B can't be subtracted" << endl;

}

break;

case '4':

if (isMultiply)

{

result = A \* B;

cout << "Multiplication Result" << endl;

cout << result;

}

else

{

cout << "A and B can't be multiplied" << endl;

}

break;

case '5':

cout << "Transpose of A" << endl;

A.printTranspose();

cout << "Transpose of B" << endl;

B.printTranspose();

break;

default:

cout << "Invalid choice" << endl;

break;

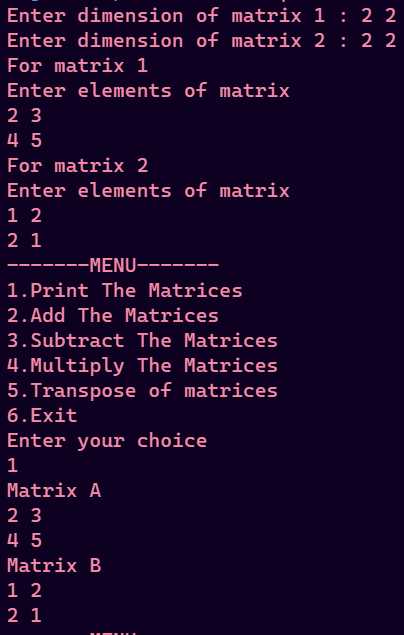
}

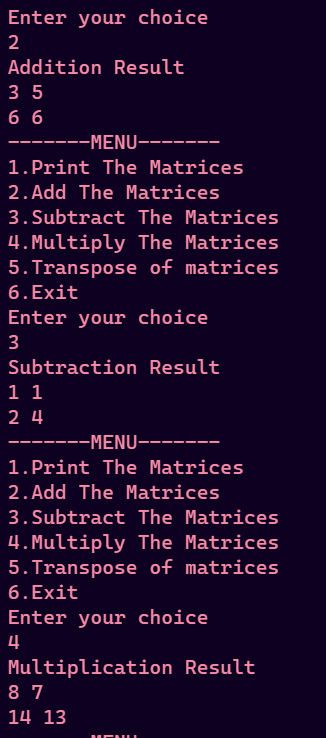
}

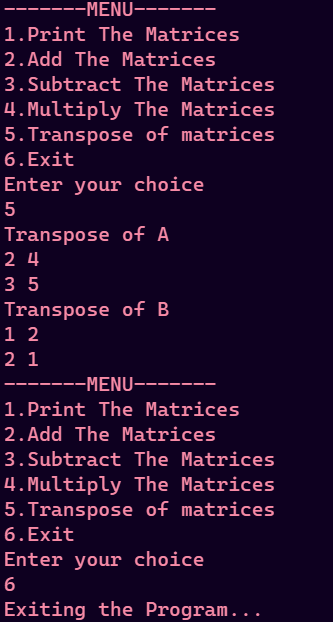
return 0;

}

**OUTPUT**

****

****

****

**Q23)**

**SOLUTION**

#include <iostream>

#include <cmath>

using namespace std;

class Triangle{

float a , b , c, base , height;

public:

Triangle(float v1 , float v2 , float v3){

a = v1;

b = v2;

c = v3;

base = 0;

height = 0;

}

Triangle(float v1 , float v2){

a = 0;

b = 0;

c = 0;

base = v1;

height = v2;

}

float getArea(){

if (a != 0){

float s = (a + b + c)/2.0;

float area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

return area;

}

return 0.5 \* base \* height;

}

};

int main()

{

Triangle t1 = Triangle(9 , 9 , 9);

Triangle t2 = Triangle(3 ,2.4);

cout << t1.getArea() << “ sq units “ << endl;

cout << t2.getArea() << “ sq units “<< endl;

return 0;

}

**OUTPUT**

****

**Q24)**

**SOLUTION**

#include <iostream>

using namespace std;

class Box

{

float a, b, c;

public:

Box()

{

a = 0;

b = 0;

c = 0;

}

Box(float a, float b, float c)

{

this->a = a;

this->b = b;

this->c = c;

}

float surface\_area()

{

return 2 \* (a \* b + b \* c + c \* a);

}

float volume()

{

return a \* b \* c;

}

string boxType()

{

if (a == b && a == c)

{

return "Cube";

}

return "Cuboid";

}

};

int main()

{

float a, b, c;

cout << "Enter sides of box" << endl;

cin >> a >> b >> c;

char choice;

Box box = Box(a, b, c);

while (true)

{

cout << "------MENU------" << endl;

cout << "1.Surface Area" << endl;

cout << "2.Volume" << endl;

cout << "3.Is cuboid/cube" << endl;

cout << "4.Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

if (choice == '4')

{

cout << "Exiting program..." << endl;

break;

}

else if (choice == '1')

{

cout << box.surface\_area() << " sq units." << endl;

}

else if (choice == '2')

{

cout << box.volume() << " cu units." << endl;

}

else if (choice == '3')

{

cout << "Type: " << box.boxType() << endl;

}

else

{

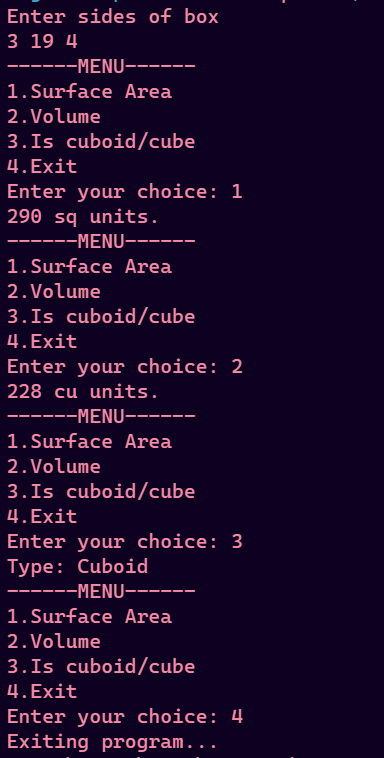
cout << "Enter a valid " << endl;

}

}

return 0;

}

**OUTPUT  
**