Q1. Write a program to compute the sum of the first n terms of the following series:

The number of terms n is to be taken from the user through the command line. If the

command line argument is not found then prompt the user to enter the value of n.

#include <iostream>

#include <cmath>

using namespace std;

int main(int argc, char \*argv[])

{

int n;

if (argc >= 2)

{

n = atoi(argv[1]);

}

else

{

cout << "enter the no of the term :";

cin >> n;

};

float sum = 0;

if (n > 0)

{

for (int i = 1; i <= n; i++)

{

sum += pow(-1, i + 1) / pow(i, i);

}

}

cout << "S = " << sum;

return 0;

}

OUTPUT:

Command line output:

PAGE 2

Executable output:

Q2. Write a program to remove duplicates from an array.

#include <iostream>

using namespace std;

int main()

{

int size;

cout << "Enter size of array :";

cin >> size;

int arr[size];

int element;

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

{

cout << "Enter element " << i + 1 << " : ";

cin >> arr[i];

}

cout << "Entered array : " << endl;

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

{

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

PAGE 3

}

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

{

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

{

if (arr[i] == arr[j])

{

for (int k = j; k < size - 1; k++)

{

arr[k] = arr[k + 1];

}

size--;

j--;

}

}

}

cout << endl

<< "After removing duplicates : " << endl;

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

{

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

}

return 0;

}

OUTPUT:

Q3. Write a program that prints a table indicating the number of occurrences of each

alphabet in the text entered as command line arguments.

#include <iostream>

#include <map>

using namespace std;

int main(int argc, char \*argv[])

PAGE 4

{

map<char, int> alphabetCount;

for (int i = 1; i < argc; i++)

{

for (int j = 0; argv[i][j] != '\0'; j++)

{

char ch = argv[i][j];

if (isalpha(ch))

{

alphabetCount[ch]++;

}

}

}

for (char ch = 'A'; ch <= 'z'; ch++)

{

if (alphabetCount[ch])

{

cout << ch << ": " << alphabetCount[ch] << endl;

}

}

return 0;

}

OUTPUT:

Q4. Write a menu driven program to perform string manipulation (without using inbuilt

string functions):

a. Show address of each character in string

b. Concatenate two strings.

c. Compare two strings

PAGE 5

d. Calculate length of the string (use pointers)

e. Convert all lowercase characters to uppercase

f. Reverse the string

g. Insert a string in another string at a user specified position

#include <iostream>

using namespace std;

void showAddress(const char\* str) {

for (int i = 0; str[i] != '\0'; i++) {

cout << "Address of " << str[i] << ": " << (void\*)&str[i] << endl;

}

}

int calculateLength(const char\* str) {

int length = 0;

while (\*str != '\0') {

length++;

str++;

}

return length;

}

void concatenateStrings(char\* str1, const char\* str2) {

int len1 = calculateLength(str1);

int len2 = calculateLength(str2);

for (int i = 0; i < len2; i++) {

str1[len1 + i] = str2[i];

}

str1[len1 + len2] = '\0';

}

int compareStrings(const char\* str1, const char\* str2) {

int i = 0;

while (str1[i] == str2[i]) {

if (str1[i] == '\0')

return 0;

i++;

}

return str1[i] - str2[i];

}

void convertToUppercase(char\* str) {

for (int i = 0; str[i] != '\0'; i++) {

if (str[i] >= 'a' && str[i] <= 'z') {

PAGE 6

str[i] -= 32;

}

}

}

void reverseString(char\* str) {

int len = calculateLength(str);

for (int i = 0, j = len - 1; i < j; i++, j--) {

char temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

void insertString(char\* str1, const char\* str2, int position) {

int len1 = calculateLength(str1);

int len2 = calculateLength(str2);

for (int i = len1 - 1; i >= position; i--) {

str1[i + len2] = str1[i];

}

for (int i = 0; i < len2; i++) {

str1[position + i] = str2[i];

}

str1[len1 + len2] = '\0';

}

int main() {

char str1[100], str2[50];

int choice, position;

cout << "Enter the first string: ";

cin>>str1;

while (true) {

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

cout << "1. Show address of each character in the string\n";

cout << "2. Concatenate two strings\n";

cout << "3. Compare two strings\n";

cout << "4. Calculate the length of the string\n";

cout << "5. Convert all lowercase characters to uppercase\n";

cout << "6. Reverse the string\n";

cout << "7. Insert a string in another string at a user-specified

position\n";

cout << "8. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

showAddress(str1);

PAGE 7

break;

case 2:

cout << "Enter the second string: ";

cin>>str2;

concatenateStrings(str1, str2);

cout << "Concatenated string: " << str1 << endl;

break;

case 3:

cout << "Enter the second string: ";

cin>>str2;

if (compareStrings(str1, str2) == 0)

cout << "Strings are equal.\n";

else

cout << "Strings are not equal.\n";

break;

case 4:

cout << "Length of the string: " << calculateLength(str1)

<< endl;

break;

case 5:

convertToUppercase(str1);

cout << "String after converting lowercase characters to

uppercase: " << str1 << endl;

break;

case 6:

reverseString(str1);

cout << "Reversed string: " << str1 << endl;

break;

case 7:

cout << "Enter the string to insert: ";

cin>>str2;

cout << "Enter the position to insert (0 - " <<

calculateLength(str1) << "): ";

cin >> position;

if (position < 0 || position > calculateLength(str1))

cout << "Invalid position.\n";

else {

insertString(str1, str2, position);

cout << "Modified string: " << str1 << endl;

}

break;

case 8:

cout << "Exiting program...\n";

return 0;

PAGE 8

default:

cout << "Invalid choice. Please try again.\n";

}

}

}

OUTPUTS:

PAGE 9

Q5. Write a program to merge two ordered arrays to get a single ordered array.

#include <iostream>

using namespace std;

int main()

{

int size1, size2;

cout << "Enter the size of the first array: ";

cin >> size1;

int arr1[size1];

cout << "Enter the elements of the first array in ascending order: ";

for (int i = 0; i < size1; i++)

{

cin >> arr1[i];

PAGE 10

}

cout << "Enter the size of the second array: ";

cin >> size2;

int arr2[size2];

cout << "Enter the elements of the second array in ascending order: ";

for (int i = 0; i < size2; i++)

{

cin >> arr2[i];

}

int mergedSize = size1 + size2;

int mergedArr[mergedSize];

int a = 0, b = 0, c = 0;

while (a < size1 && b < size2)

{

if (arr1[a] < arr2[b])

{

mergedArr[c++] = arr1[a++];

}

else

{

mergedArr[c++] = arr2[b++];

}

}

while (a < size1)

{

mergedArr[c++] = arr1[a++];

}

while (b < size2)

{

mergedArr[c++] = arr2[b++];

}

cout << "Merged array in ascending order: ";

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

{

cout << mergedArr[i] << " ";

}

return 0;

}

OUTPUT:

PAGE 11

Q6. Write a program to search a given element in a set of N numbers using Binary

search

(i) with recursion

(ii) without recursion.

#include <iostream>

using namespace std;

int binarySearchRecursive(int arr[], int low, int high, int target) {

if (low > high)

return -1;

int mid = low + (high - low) / 2;

if (arr[mid] == target)

return mid;

else if (arr[mid] > target)

return binarySearchRecursive(arr, low, mid - 1, target);

else

return binarySearchRecursive(arr, mid + 1, high, target);

}

int binarySearchIterative(int arr[], int size, int target) {

int low = 0;

int high = size - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == target)

return mid;

else if (arr[mid] > target)

high = mid - 1;

else

low = mid + 1;

}

return -1;

}

int main() {

int size, target;

PAGE 12

cout << "Enter the size of the array: ";

cin >> size;

int arr[size];

cout << "Enter the elements of the array in ascending order: ";

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

cin >> arr[i];

}

cout << "Enter the element to search: ";

cin >> target;

int recursiveIndex = binarySearchRecursive(arr, 0, size - 1, target);

if (recursiveIndex != -1)

cout << "Element found at index (using recursion): " <<

recursiveIndex << endl;

else

cout << "Element not found (using recursion).\n";

int iterativeIndex = binarySearchIterative(arr, size, target);

if (iterativeIndex != -1)

cout << "Element found at index (without recursion): " <<

iterativeIndex << endl;

else

cout << "Element not found (without recursion).\n";

return 0;

}

OUTPUT:

Q7. Write a program to calculate GCD of two numbers (i) with recursion (ii) without

recursion

#include <iostream>

using namespace std;

int gcdRecursive(int a, int b)

{

if (b == 0)

return a;

PAGE 13

return gcdRecursive(b, a % b);

}

int gcdIterative(int a, int b)

{

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int main()

{

int num1, num2;

cout << "Enter the first number: ";

cin >> num1;

cout << "Enter the second number: ";

cin >> num2;

int recursiveGCD = gcdRecursive(num1, num2);

cout << "GCD (using recursion): " << recursiveGCD << endl;

int iterativeGCD = gcdIterative(num1, num2);

cout << "GCD (without recursion): " << iterativeGCD << endl;

return 0;

}

OUTPUT:

Q8. Create a Matrix class. Write a menu-driven program to perform following Matrix

operations (exceptions should be thrown by the functions if matrices passed to them

are incompatible and handled by the main() function):

a. Sum

b. Product

c. Transpose

PAGE 14

#include <iostream>

using namespace std;

class Matrix

{

int row, col;

public:

int A[5][5];

void matrix\_input(int m)

{

cout << "Enter the size of matrix " << m << " :";

int r, c;

cin >> r >> c;

this->row = r;

this->col = c;

cout << "Enter the matrix " << m << " : " << endl;

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

{

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

{

cin >> A[i][j];

}

}

}

void matrix\_display()

{

int i, j;

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

{

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

{

cout << A[i][j] << ",";

}

cout << endl;

}

}

bool matrix\_add\_check(Matrix o2)

{

if ((row != o2.row) or (col != o2.col))

{

return false;

}

else

{

PAGE 15

return true;

}

}

Matrix matrix\_add(Matrix o2)

{

Matrix addition\_matrix;

addition\_matrix.row = row;

addition\_matrix.col = col;

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

{

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

{

addition\_matrix.A[i][j] = A[i][j] + o2.A[i][j];

}

}

return addition\_matrix;

}

bool matrix\_multiply\_check(Matrix o2)

{

if (col != o2.row)

{

return false;

}

else

{

return true;

}

}

Matrix matrix\_multiply(Matrix o2)

{

Matrix multiplication\_matrix;

multiplication\_matrix.row = row;

multiplication\_matrix.col = o2.col;

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

{

for (int j = 0; j < o2.col; j++)

{

int mul = 0;

for (int k = 0; k < col; k++)

{

mul = mul + A[i][k] \* o2.A[k][j];

PAGE 16

}

multiplication\_matrix.A[i][j] = mul;

}

}

return multiplication\_matrix;

}

Matrix matrix\_transpose()

{

Matrix transpose\_matrix;

transpose\_matrix.row = col;

transpose\_matrix.col = row;

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

{

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

{

transpose\_matrix.A[j][i] = A[i][j];

}

}

return transpose\_matrix;

}

};

int main()

{

Matrix m1;

m1.matrix\_input(1);

Matrix m2;

m2.matrix\_input(2);

cout << "========================\nThe m1 matrix is : " << endl;

m1.matrix\_display();

cout << "The m2 matrix is : " << endl;

m2.matrix\_display();

int menu\_option = 0;

while (1)

{

cout << "==========MENU==========\n0) Exit. \n1) Sum of the

matrices.\n2) Product of the matrices.\n3) Transpose of the

matrices.\nEnter a choice (0/1/2/3) : ";

cin >> menu\_option;

switch (menu\_option)

{

case (0):

PAGE 17

cout << "====================\nThanks!";

exit(0);

break;

case (1):

try

{

if (m1.matrix\_add\_check(m2))

{

cout << "====================\nSum of matrices is

:\n";

((m1).matrix\_add(m2)).matrix\_display();

}

else

{

throw "Matrices size mismatch! Addition is not

possible.";

}

}

catch (const char \*p)

{

cout << "Error! " << p << endl;

}

break;

case (2):

try

{

if (m1.matrix\_multiply\_check(m2))

{

cout << "====================\nProduct of matrices is

:\n";

((m1).matrix\_multiply(m2)).matrix\_display();

}

else

{

throw "Matrices size mismatch! Multiplication is not

possible.";

}

}

catch (const char \*p)

{

cout << "Error! " << p << endl;

}

break;

PAGE 18

case (3):

cout << "====================\nTranspose of the matrices is

:\nm1 transpose : \n";

m1.matrix\_transpose().matrix\_display();

cout << "m2 transpose : \n";

m2.matrix\_transpose().matrix\_display();

break;

}

}

return 0;

}

OUTPUT:

PAGE 19

Q9. Define a class Person having name as a data member. Inherit two classes Student

and Employee from Person. Student has additional attributes as course, marks and

year and Employee has department and salary. Write display() method in all the three

classes to display the corresponding attributes. Provide the necessary methods to

show runtime polymorphism.

#include <iostream>

using namespace std;

class Person

{

public:

string name;

Person(string \_name)

{

name = \_name;

PAGE 20

}

virtual void display()

{

cout << "Name : " << name << endl;

}

};

class Student : public Person

{

public:

string course;

float marks;

int year;

Student(string \_name, string \_course, float \_marks, int \_year) :

Person(\_name)

{

name = \_name;

course = \_course;

marks = \_marks;

year = \_year;

}

void display()

{

cout << "Name : " << name << endl;

cout << "Course : " << course << endl;

cout << "Marks : " << marks << endl;

cout << "Year : " << year << endl;

}

};

class Employee : public Person

{

public:

string department;

float salary;

Employee(string \_name, string \_department, float \_salary) :

Person(\_name)

{

name = \_name;

department = \_department;

salary = \_salary;

}

void display()

{

cout << "Name : " << name << endl;

PAGE 21

cout << "Department : " << department << endl;

cout << "Salary : " << salary << endl;

}

};

int main()

{

Person \*bptr;

Student s1("Ajit", "BSC", 300, 2022);

bptr = &s1;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

bptr->display();

Employee e1("Anay", "Sales", 120000);

bptr = &e1;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

bptr->display();

return 0;

}

OUTPUT:

Q10. Create a Triangle class. Add exception handling statements to ensure the following

conditions: all sides are greater than 0 and sum of any two sides are greater than the

third side. The class should also have overloaded functions for calculating the area of a

right angled triangle as well as using Heron's formula to calculate the area of any type

of triangle.

#include <iostream>

#include <cmath>

using namespace std;

class Triangle

{

float side1, side2, side3;

public:

PAGE 22

Triangle(float \_side1, float \_side2, float \_side3)

{

try

{

if (!(\_side1 > 0 && \_side2 > 0 && \_side3 > 0))

{

throw "Triangle side length cannot be 0 or less.";

}

if (!(((\_side1 + \_side2) > \_side3) && ((\_side2 + \_side3) >

\_side1) && ((\_side3 + \_side1) > \_side2)))

{

throw "Sum of two sides should be greater than the third

side in a triangle.";

}

side1 = \_side1;

side2 = \_side2;

side3 = \_side3;

if (side1 \* side1 + side2 \* side2 == side3 \* side3)

{

cout << "Area of right-angled triange with sides (" <<

side1 << "," << side2 << "," << side3 << ") is : " << area(side1, side2)

<< endl;

}

else if (side2 \* side2 + side3 \* side3 == side1 \* side1)

{

cout << "Area of right-angled triange with sides (" <<

side1 << "," << side2 << "," << side3 << ") is : " << area(side2, side3)

<< endl;

}

else if (side3 \* side3 + side1 \* side1 == side2 \* side2)

{

cout << "Area of right-angled triange with sides (" <<

side1 << "," << side2 << "," << side3 << ") is : " << area(side3, side1)

<< endl;

}

else

{

cout << "Area of non right-angled triangle with sides ("

<< side1 << "," << side2 << "," << side3 << ") is : " << area(side1,

side2, side3) << endl;

}

}

catch (const char \*s)

{

PAGE 23

cout << "Error! " << s << endl;

}

}

float area(float base, float height)

{

return (0.5 \* base \* height);

}

float area(float s1, float s2, float s3)

{

float s = (s1 + s2 + s3) / 2;

return (sqrt(s \* (s - s1) \* (s - s2) \* (s - s3)));

}

};

int main()

{

Triangle t1(3, 4, 5);

Triangle t2(4, 4, 5);

Triangle t3(0, 0, 0);

Triangle t4(2, 2, 4);

return 0;

}

OUTPUT:

Q11. Create a class Student containing fields for Roll No., Name, Class, Year and Total

Marks. Write a program to store 5 objects of Student class in a file. Retrieve these

records from the file and display them.

#include <iostream>

#include <fstream>

using namespace std;

class Student {

public:

int rollNo;

string name;

string className;

int year;

int totalMarks;

PAGE 24

void getData() {

cout << "Enter Roll No.: ";

cin >> rollNo;

cout << "Enter Name: ";

cin.ignore();

getline(cin, name);

cout << "Enter Class: ";

getline(cin, className);

cout << "Enter Year: ";

cin >> year;

cout << "Enter Total Marks: ";

cin >> totalMarks;

}

void displayData() {

cout << "Roll No.: " << rollNo << endl;

cout << "Name: " << name << endl;

cout << "Class: " << className << endl;

cout << "Year: " << year << endl;

cout << "Total Marks: " << totalMarks << endl;

}

};

int main() {

const int numStudents = 5;

Student students[numStudents];

ofstream outFile("students.txt");

if (!outFile) {

cerr << "Error opening file." << endl;

return 1;

}

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

cout << "Enter details for Student " << i + 1 << ":" << endl;

students[i].getData();

outFile << students[i].rollNo << endl;

outFile << students[i].name << endl;

outFile << students[i].className << endl;

outFile << students[i].year << endl;

outFile << students[i].totalMarks << endl;

outFile << endl;

}

outFile.close();

ifstream inFile("students.txt");

PAGE 25

if (!inFile) {

cerr << "Error opening file." << endl;

return 1;

}

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

cout << "Details of Student " << i + 1 << ":" << endl;

inFile >> students[i].rollNo;

inFile.ignore();

getline(inFile, students[i].name);

getline(inFile, students[i].className);

inFile >> students[i].year;

inFile >> students[i].totalMarks;

inFile.ignore();

students[i].displayData();

cout << endl;

}

inFile.close();

return 0;

}

OUTPUT:

PAGE 26

CREATED FILE:

OUTPUT FROM FILE:

PAGE 27

Q12. Copy the contents of one text file to another file, after removing all whitespaces.

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

string inputFile, outputFile;

cout << "Enter the name of the input file: ";

getline(cin, inputFile);

PAGE 28

cout << "Enter the name of the output file: ";

getline(cin, outputFile);

ifstream inFile(inputFile);

if (!inFile)

{

cerr << "Error opening input file." << endl;

return 1;

}

ofstream outFile(outputFile);

if (!outFile)

{

cerr << "Error opening output file." << endl;

return 1;

}

char ch;

while (inFile.get(ch))

{

if (!isspace(ch))

{

outFile << ch;

}

}

cout << "File copied successfully after removing whitespaces." <<

endl;

inFile.close();

outFile.close();

return 0;

}