Advanced Computer Programming (4.17) Class Test

3 May 2018

**Answer All 4 questions in Part A and**

**2 out of the 3 questions in part B**

**Total Marks: 80**

Total time for test: 3hrs

**Section A: (Answer all 4 questions) (10 marks each)**

**Question A.1**

Obtain a floating point number from the user. Ensure that the number is between -1 and 1 by obtaining the user input within a loop, with the user being asked for a new number until an appropriate one is provided. Calculate the arcsine of this number and display the answer in both radians and degrees (note that the asin function only returns the answer in radians and therefore you must convert it to degrees).

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

using namespace std;

double theta;

void main(void)

{

cout << "Enter a number between 1 and -1 for a sine value:" << endl;

cin >> theta;

if (theta>=-1 && theta <=1)

{

cout << "The sine value you have entered leads to an angle of: " << asin(theta) << " radians,"<<endl;

cout << "or in: " << asin(theta)\*360/M\_PI << " degrees." << endl;

}

else (cout << "The value you have entered is not between 1 and -1!!!");

}

**Question A.2**

Write a program to calculate the value of the constant *e*. This can be done based on the Taylor series expansion:  
where is the factorial of (i.e. )

Use a loop to calculate this value. This series converges quickly and so you need only go out to n = 20. The program should display both this calculated value of *e*, as well as that obtained using exp(1.0). Note that the factorial need not be calculated from scratch for each iteration of the loop as

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

using namespace std;

double e = 1.0;

float find1overfactorial(int n)

{

if (n == 0) return(1.0);

else if (n == 1) return(1.0);

else return(1.0 / n \* find1overfactorial(n - 1));

}

void main(void)

{

for (int i = 1; i <= 20; i++)

{

cout << find1overfactorial(i) <<endl;

e += find1overfactorial(i);

}

cout << "The e value is: " << e <<endl;

cout << "The actual value is: " << exp(1.0) << endl;

}

**Question A.3**

Write a **function** that takes in a string and then checks if the string is a palindrome, returning true if it is a palindrome, otherwise returning false. A palindrome is a word or sentence that is spelt the same forwards or backwards (e.g. "anna", "now I won" or "able was I ere I saw elba"). To do this you need to loop over half the string and check if the *i*th character is the same as the (*string length-1-i*)th character (note that the -1 is because C/C++ arrays are zero based).

In the main function read in the string from the user and use the function to check if the string is a palindrome, writing an appropriate comment to screen.

#include <iostream>

#include <cstring>

using namespace std;

char input[100];

bool palin(char \*str)

{

for (int i = 0; i < strlen(str); i++)

if (input[i] != input[strlen(str) - 1 - i])

return false;

return true;

}

void main(void)

{

cout << "Enter your sentence" << endl;

cin.getline(input,100);

if (palin(input)) cout << input << " is a Palindrome" << endl;

else cout << input << "is notb a palindrome"<< endl;

}

**Question A.4**

Write a **recursive** function to find the greatest common divisor of 2 integers (e.g. the greatest common divisor of 20 and 8 is 4). The recursive way to do this is by using something known as the Euclidian algorithm. The algorithm can be expressed as follows:

The greatest common devisor of x and y is y if the remainder when x is divided by y is zero, otherwise it is the greatest common devisor of y and the remainder when x is divided by y (remember that the remainder is obtained using the % operator).

Allow the user to enter a pair of numbers and display the greatest common divisor.

#include <iostream>

using namespace std;

int gcd(int x, int y)

{

if (x%y == 0) return y;

else return gcd(y, x%y);

}

void main(void)

{

int n, m;

cout << "Enter a No.: " << endl;

cin >> n;

cout << "Enter another No.: " << endl;

cin >> m;

cout << "Their GCD is " << gcd(n, m) << endl;

}

**Section B: (Answer 2 out of 3 questions only) (20 marks each)**

**Question B.1**

Create a class which stores the name of a location together with its latitude and longitude. Write a member function that reads in the information from the user, with the latitude and longitude being obtained in degrees, but then also stored as radians (i.e. two member variables for both latitude and longitude). Write another member function which displays the name and location in both radians and degrees.

Write a further member function which takes in an objects of the class you have created and which returns the great circle distance from the original object to the object being passed to the function. You can use the following equations to calculate this distance:

Where is the latitude, is the longitude and is the radius of the earth (6 371 km). Note that returns the inverse tan, but does so based on the opposite and adjacent edge lengths rather than their ratio (this ensures that the quadrant for the angle is correct). Also note that the sines are squared.

In the main body of the code allow the user to specify how many locations they wish to enter and then use “new” to dynamically allocate an array of the location objects. The user should then be able to enter the details for each of the objects using the appropriate member function.

The code must then display each possible pair of locations, using the member function to display the names and coordinates, together with the distance between them. To avoid duplicating pairs you should loop over all the locations and then within that loop you should loop over all locations with an index higher than that of the first location being considered.

Remember to free the memory before the program exits.

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

#include <cstring>

#include <conio.h>

using namespace std;

double R = 6371000.0;

class Cloc

{

public:

double phi, lambda;

char name[100];

Cloc();

double alpha (Cloc loc2);

void input(void);

};

Cloc::Cloc()

{

for (int i = 0; i < 100; i++) name[i] = 0;

phi = 0.0;

lambda = 0.0;

}

double Cloc::alpha(Cloc loc2)

{

return pow(sin((loc2.phi\* M\_PI / 180 - phi \* M\_PI / 180)/2), 2)

+ cos(phi\* M\_PI / 180)\*cos(loc2.phi\* M\_PI / 180)

\*pow(sin((loc2.lambda\* M\_PI / 180 - lambda \* M\_PI / 180)/0.5), 2);

}

void Cloc::input(void)

{

cout << "Name of Location: ";

cin.ignore();

cin.getline(name, 100);

cout << "Latitude in degrees: ";

cin >> phi;

cout << "Longitude in degrees: ";

cin >> lambda;

}

double theta(double a)

{

return 2 \* atan(sqrt(a) / sqrt(1 - a));

}

double distance(double a, double b)

{

return a \* b;

}

Cloc \*all;

char name[100];

void main(void)

{

int np;

cout << "No. of places to put in: ";

cin >> np;

all = (Cloc \*) new Cloc\*[np];

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

{

all[i] = Cloc();

all[i].input();

cout << endl;

}

int m;

cout << "Choose a point: ";

for (int k = 0; k < np; k++) cout << k << " " << all[k].name << endl;

cout << endl;

cin >> m;

cout << endl;

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

{

if (j!=m)

{

cout << "The distance from " << all[m].name << " to " << all[j].name << " is:" << endl;

cout << "To " << all[j].name << " is: " << endl;

cout << distance(theta(all[m].alpha(all[j])), R);

}

}

delete all;

}

**Question B.2**

A Perfect number is one where the sum of the factors of that number add up to the number itself. For instance 28 is a Perfect number as its factors are 1, 2, 4, 7 and 14, which add up to 28. Write a computer program that finds and displays all the Perfect numbers below 10 000 (there are only 4 of them) and also displays their factors. The code should also write this data to file.

To find the Perfect numbers you must test every number less than 10 000. For each number being tested you must find all of its factors, which involves checking all the numbers less than the one being tested to see if they are a factor (i.e. have a remainder of zero when the number being tested is divided by this number). All these factors should be summed and the sum then checked to see if it adds up to the number being tested.

To display the factors you must store them in an array as they are found, but only print them out if the number being tested is a Perfect number (note that there are other ways to display the factors, but this is the method that I wish you to use).

Remember to also write all the Perfect numbers found, as well as their factors, to a single text file.

#include <iostream>

#include <fstream>

using namespace std;

int x[4], s=0;

fstream myFile;

void main(void)

{

for (int i=2; i < 10001; i++)

{

int sumofdiv = 0;

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

if(i%j==0) sumofdiv += j;

if (sumofdiv == i)

{

x[s] = i;

s++;

}

}

myFile.open("2018examQB2.txt", fstream::out);

if (myFile.fail())

{

cout << "Error opening file!!" << endl;

exit(0);

}

for (int i = 0; i < 4; i++)

{

cout << x[i] << ": ";

myFile << x[i] << " is a Perfect Number, with factors: " << "\t";

for (int j = 1; j < x[i]; j++) if (x[i] % j == 0)

{

cout << j << " ";

myFile << j << " & ";

}

cout << endl;

myFile << "\t";

}

}

**Question B.3**

Write a program to calculate the standard deviation of a list of numbers. The numbers should be stored as a **linked list**. The program must **NOT** ask how many items are to be added beforehand, but should instead add numbers to the list until the user indicates that they do not wish to add any more numbers.

Write a **function** that takes in a pointer to the first item in the list and returns the standard deviation of the list. Use the following formula for the standard deviation, using a **loop** to obtain the required sums (*N* being the number of items in the list and being the *i*th item, though as this is a linked list, the items won’t actually have an index):

The program should use this function to calculate the standard deviation of the list and display it.

Once the program has displayed the standard deviation of the list, the user should be given the option to add more numbers to the list, clear the list and add new numbers or to exit the program. Remember to free all memory before the program exits.