DEPARTMENT OF CHEMICAL ENGINEERING

SUBJECT – COMPUTER APPLICATION IN Ch.ENGG LABORATORY

NAME OF GROUP MEMBER	ROLL NUMBER
ABHISHIKTA BISWAS	002110301155
RAHUL GHOSH	002110301156
ANISH SARKAR	002110301157
ARKA PRABHO SOME	002110301158
BISWAYAN PAUL	002110301160
GARBA GOSWAMI	002110301161
ROHIT BARUA	002110301200
PRATIBHAS PATRA	002110301202

QUESTION

Write a program to obtain ln(1.1) correct to fifth decimal place using infinite series expression for ln(1+x).

ALGORITHM:

Step 1: Initiating a variable x with value 0.1.

Step 2: Initiating another variable prevans with value 0.

Step 3: A variable ans is defined to store the answer in each step starting from 0.

Step 4: A variable prevans is defined to store the answer in previous step starting from 0.

Step 5: Creating a For loop for calculating in each expansion term of the Taylor series of ln(1+x) format.

Step 5.1: For loop is initiated with value i. For loop is calculated until number of iterations becomes 100.

Step 5.2: prevans is assigned the value of ans.

Step 5.3: i is checked for even or odd and accordingly the value is subtracted/added for each term in the Taylor series respectively.

Step 5.4: Another if condition is given to check the difference between previous value and present value. If it is same up to 5 decimal places i.e. difference less than 10^{-5}, the for loop is broken and the step comes outside of the loop.

Step 5.5: If 5.4 is not satisfied i is updated with i = i+1, return to step 5.2.

Step 6: Outputting the value of ln (1.1) i.e., ln (1+0.1) with tolerance 5 decimal place and maximum iteration 100 whichever is achieved earlier.

CODE:

```
#include <bits/stdc++.h>

using namespace std;

int main()
```

```
{

float x = 0.1;
float prevans = 0;
float ans = 0;
for (int i = 1; i <= 100; i++)
{

prevans = ans;
if (i % 2 == 0)
{

ans = ans - float(pow(x, i) / i);
}

else
{

ans = ans + float(pow(x, i) / i);
}

if (abs(ans - prevans) < 1e-5)
{

break;
}
}

//Correcting the answer upto 5 Decimal places by explicit type casting
ans =  ( (int) (ans/(1e-5)) ) * (1e-5) ;

cout << "The value of ln(1+0.1) corrected up to 5 decimal places is = " <<ans;
return 0;

}
```

OUTPUT:

```
The value of ln(1+0.1) corrected upto 5 decimal places is = 0.09531
```

DISCUSSION:

We know that the expansion of ln(1+x) is given by Taylor’s series which come out to be

$$\mathbf{ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} \ldots \ldots \ldots ..}$$

We were required to find out ln (1+0.1) up to a precision of 5 decimal places. We have ensured this by keeping a condition of checking the value of increment/decrement of each term of Taylor series. If the value of the increment/decrement is less that 10^{-5}, then we have terminated the loop. Hence achieving the target of 5 decimal places. We have also introduced a maximum iteration of 100 of the loops. In case the desired loop is not achieved so this condition will ensure the loop does not enter a state of infinite iterations and the value is printed accordingly.

QUESTION

Solve the following equation by Newton-Raphson method: cos(x)-x exp(x)=0

ALGORITHM

Step1: -The function is defined using a function name ”func”.

Step-2: -The differentiation of the function is defined using a user-defined function called “diff”.

Step-3: -Initial value(X_0) is assumed to be 0.

Step-4: - A variable oldsign is used to check the sign of the function at the value of X_0=0 with 1 for positive and -1 for negative.

Step-5: -Another variable newsign is used to find the sign of the function at any X_o with X_0 value starting from 0.

Step-6: -A while loop is being constructed.

Step-6.1: -Old sign and new sign are being checked if they are of the same sign.

Step-6.2: - The value of X_0 is updated by 0.1, oldsign is given the previous value of newsign, newsign is found again using the user-defined function func and we again start from the step-7.

Step-6.3: -If not the loop ends (or exits).

Step-7: - A variable maxItr is defined for the maximum number of iterations, which is taken as 1000 here.

Step-8: -A variable tolX is defined to find the tolerance of error, which is taken as 1e-6 (or 10^{-6}) here.

Step-9: - A variable x is assigned different values of the possible roots in each iteration, starting from X_o.

Step-10:- Another variable xOld is used to store the value of root from previous iteration for the purpose of checking precision.

Step-11: - A for loop is constructed to find the root using Newton Raphson technique.

Step-11.1: - X is updated in each iteration using the value of xOld (previous iteration) and its corresponding function value from “func” named function and differentiated function value form “diff” named function.

Step-11.2: -A variable err is defined to check the absolute difference between the X and xOld.

Step-11.3: -If block is used to check if the tolerance is below the required tolerance (10^{-6}) then the loop exits, else back to step 11.

Step-12: - The value of X is being displayed on the terminal screen at the output.

Step-13: -The value of function func at this value of X is also displayed in the output window.

CODE

```
#include <bits/stdc++.h>
using namespace std;

// f(x)  = cos(x)-x*exp(x)
// f'(x) = -sin(x) - {exp(x) + x*exp(x)}

double func(double x)
{
   return cos(x) - (x * exp(x));
}

double diff(double x)
{
   return -sin(x) - (exp(x) + (x * exp(x)));
}

int main()
{
   double x0 = 0; // initial guess
  double newsign = func(x0)/abs(func(x0));
  double oldsign = func(x0)/abs(func(x0));
   while ((oldsign*newsign)>0)
```

```
    {
        oldsign = newsign;

        x0 += .1;
        newsign = func(x0)/abs(func(x0));
    }

    double maxItr = 1000;
    double tolX = 1e-6;

    double x = x0;     double xOld = x0;

    for (int i = 1; i <= maxItr; i++)
    {
        double f = func(x);  // returns f(x)
        double df = diff(x); // returns f'(x)

        x = xOld - (f / df); // finding new x

        double err = abs(x - xOld); // error calculation

        if (err < tolX) // checking if error is less than tolerance
        {           break;
        }
        xOld = x;
    }
    cout << "The value of x corrected upto 6 decimal places = " << x << endl;     cout << "The
corresponding value of f(x) = " << cos(x) - (x * exp(x)) << endl;     return 0;
}
```

<u>OUTPUT</u>

```
The value of x corrected upto 6 decimal places = 0.517757

The corresponding value of f(x) = 1.11022e-16
```

DISCUSSION

The Newton Raphson is one of the most widely used methods for finding the roots of any equation. It is an iterative method which gives the approximate roots for any given equation. The Newton Raphson formula is given by this :-

$$X_{(n+1)} = X_n - (f(X_n) / f'(X_n))$$

Where,
X_n gives the value of X in the nth iteration $X_{(n+1)}$ gives the value of X in the (n+1)th iteration $f(X_n)$ gives the value of the function based on the value of X

$f'(X_n)$ gives the value of the differential of the function based on the value of X

In this question we are assigned a equation $\cos(x) - xe^x=0$ and we are told to find the (approximate)roots for the equation using the Newton Raphson formula. So, we have taken the maximum precision of the roots to be 10^(-6).

Accordingly, the body of the code mainly consists of two loops, the first one is to approximately determine the initial value to start the Newton Raphson method and the next loop is used to run the required iterations for the Newton Raphson method using the initial value. The second loop continues until the it achieves the tolerance value given by us or it completes 1000 iterations whichever is achieved first.