MA202 LAB1

Name: Dipean Dasgupta ID:202151188

Task 1:Write a C-program to evaluate the value of e^x in the domain [0,1]. As discussed in the Tutorial, use the Taylor theorem to do so, and ensure that the error in the evaluation is less than 10^{-6}.

Solution Code:

```c
#include <stdio.h>
#include <math.h>
int factorial(int p){
    int fact=1;
     for(int x=1;x<=p;x++){
        fact=fact*x;
    }
     return fact;
}
double Expx(double x) {
    double result = 1;
    double term = 1;
    double x_pow = x;
    for (int i = 1; i <= 9; i++) {
        term = pow(x,i)/factorial(i);
        result += term;
    }
    return result;
}
double error(double x){
    int M=9;
 double e= pow(x,M+1)/factorial(M+1);
 return e;
}

int main() {
    double x;
    for(x=0.1;x<1;x+=0.1){
    double Calculated e^x = Expx(x);
    double e= error(x);
    printf("Calculated value of e^x: %lf\n", Calculated e^x);
    printf("Error: %.20lf\n", e);
    }
```

```
    return 0;
}
```

OUTPUT:

```
D:\Cprogramming>cd "d:\Cprogramming\" && gcc expotr.c -o expotr && "d:\Cprogramming\"expotr
Calculated value of e^x: 1.000000
Error: 0.00000000000000000000
Calculated value of e^x: 1.105171
Error: 0.00000000000000002756
Calculated value of e^x: 1.221403
Error: 0.00000000000002821869
Calculated value of e^x: 1.349859
Error: 0.00000000000162723214
Calculated value of e^x: 1.491825
Error: 0.00000000002889594356
Calculated value of e^x: 1.648721
Error: 0.00000000026911444555
Calculated value of e^x: 1.822119
Error: 0.00000000166628571429
Calculated value of e^x: 2.013753
Error: 0.00000000778426060957
Calculated value of e^x: 2.225541
Error: 0.00000002958944620811
Calculated value of e^x: 2.459603
Error: 0.00000009608643080357
Calculated value of e^x: 2.718282
Error: 0.00000027557319223986
```

Task 2. See what happens to the error when you increase the no. of terms that you have included for the evaluation. Make a semi-log plot of error as a function of no. of terms in the Taylor expansion. You can use GNUPlot or any other plotting tool.

Solution Code:

```
#include <stdio.h>
#include <math.h>


int factorial(int p){
    int fact=1;
     for(int x=1;x<=p;x++){
        fact=fact*x;
    }
     return fact;
}
double Expx(double x,int n) {
    double result = 1;
```

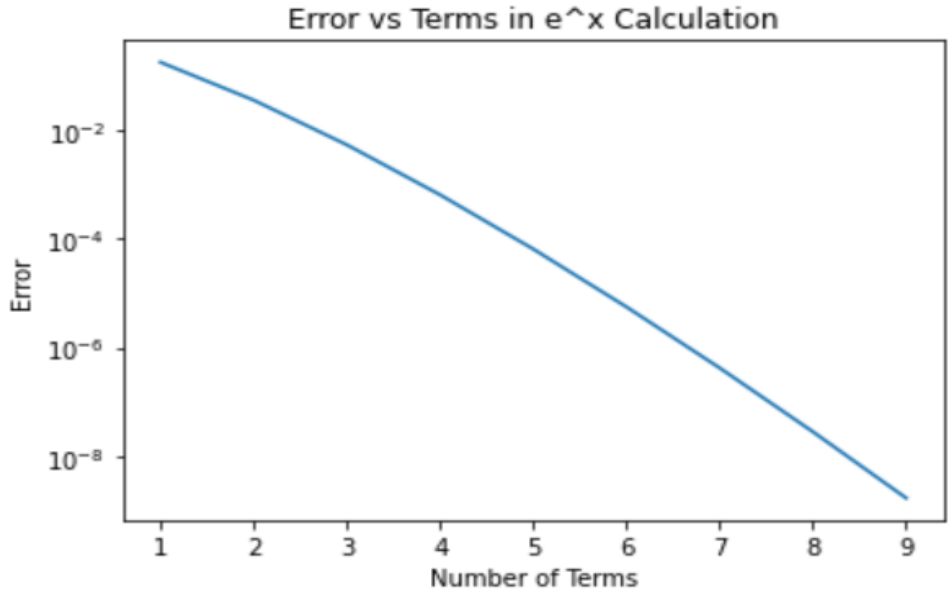
```
    double term = 1;
    double x_pow = x;
    for (int i = 1; i <= n; i++) {
        term = pow(x,i)/factorial(i);
        result += term;
    }
    return result;
}

double error(double x){
    int M=9;
    double e= pow(x,M+1)/factorial(M+1);
    return e;
}

int main() {
    double x;
    int n;
    printf("Enter a value for x: ");
    scanf("%lf", &x);
    printf("Enter number of terms:");
    scanf("%d", &n);
    double e_to_x_approx = Expx(x,n);
    double e= error(x);
    printf("Calculated value of e^x: %lf\n", e_to_x_approx);
    printf("Error: %.20lf\n", e);

    return 0;
}
```

OUTPUT:

```
D:\Cprogramming>cd "d:\Cprogramming\" && gcc
Enter a value for x: 0.6
Enter number of terms:9
Calculated value of e^x: 1.822119
Error: 0.00000000166628571429
```

Plotting the Graph:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y=
[0.17999999999999,0.03599999999999,0.00539999999999,0.000647999999999,0.00
00647999999,0.00000555428577,0.00000041657142857,0.00000002777142857,0.000
00000166628571]
plt.semilogy(x, y)
plt.xlabel('Number of Terms')
plt.ylabel('Error')
plt.title('Error vs Terms in e^x Calculation')
plt.show()
```



3. Write a C-program to evaluate the value of $tan^{-1}(x)$ in the domain $[0,1]$. You may use the following expansion: $tan^{-1}(x) = \sum_{k=0}^{M} (-1)^k \frac{x^{2k+1}}{2k+1} + R_M(x)$. Here the error term is given by $R_M(x) = (-1)^{M+1} \int_0^x dt \frac{t^{2M+2}}{1+t^2}$ which can be estimated using the fact that $\int_0^x dt \frac{t^{2M+2}}{1+t^2} \leq \int_0^x dt t^{2M+2}$. What is the value of M to be taken so as to ensure that the error in the evaluation is less than 10^{-5}. Knowing the fact that $tan^{-1}(1) = \frac{\pi}{4}$, use this program to evaluate π accurately to 10 decimal places.

Solution Code:

```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>


double Err(int M,int x){
return (pow(x,(2*M+3)))/(2*M+3);
}

double TanCalc(int M,int x){
double mx = Err(M,x);
double Total =0.0000000000;
Total+=mx;
for(int i =0;i<=M;i++){
    double Z = (pow(-1,i))*(pow(x,2*i+1))/(2*i+1);
    Total+=Z;
}
return Total;
}

int main(){
printf("The error in calculation is %lf\n",Err(50000,1));
printf("The value of tan inverse x at x = 1 and M = 50000 is
%lf\n",TanCalc(50000,1));
printf("Value of pi using M = 50000 we get %1.20f ",4*TanCalc(50000,1));
return 0;

}
```

OUTPUT:

```
D:\Cprogramming>cd "d:\Cprogramming\" && gcc tanevaluation.c -o tanevaluation && "d:\Cprogramming\"tanevaluation
The error in calculation is 0.000010
The value of tan inverse x at x = 1 and M = 50000 is 0.785413
Value of pi using M = 50000 we get 3.14165265198982090000
```