**PROGRAM NO 1**

**NAME : PARAS BHATT**

**COURSE /BRACH : BTECH(CSE)**

**SEM/SEC : 5(E)**

**ROLL NO : 40**

**DATE : 14/09/22**

**1.OBJECTIVE**

WAP TO ESTIMATE ABOSULUTE ERROR , RELATIVE ERROR AND PERCENTAGE ERROR USING THE CONCEPT OF ROUND OFF AND TRUNCATION

**2.METHOD /ALGORITHM**

**Absolute Error :**  
Let the true value of a quantity be X and the approximate value of that quantity be X1. Hence absolute error has defined the difference between X and X1. Absolute Error is denoted by EA.

Hence EA= X-X1=δX

**Relative Error :**

ER = EA/X = (Absolute Error)/X

**Percentage Error :**

EP= 100×EP= 100×EA/X

**3.PROGRAM**

#include <stdio.h>

#include <math.h>

int main()

{

double x, temp, x1;

int n;

printf("Enter the number : ");

scanf("%lf", &x);

printf("Enter the decimal place : ");

scanf("%d", &n);

temp = x \* pow(10, n);

int res = (int)(temp + 0.5);

int res1 = (int)(temp);

temp = (float)res / pow(10, n);

x1 = (float)res1 / pow(10, n);

printf("After round off we have : %lf\n", temp);

printf("After truncate we have : %lf\n", x1);

printf("absolute error %f \n", fabs(x - x1));

printf("relative error %f \n", fabs((x - x1) / x));

printf("percentage error %f \n", fabs((x - x1) / x) \* 100);

}

**4.OUTPUT :**

Enter the number : 4.543432343

Enter the decimal place : 4

After round off we have : 4.543400

After truncate we have : 4.543400

absolute error 0.000032

relative error 0.000007

percentage error 0.000712

**PROGRAM NO 2**

**NAME : PARAS BHATT**

**COURSE /BRACH : BTECH(CSE)**

**SEM/SEC : 5(E)**

**ROLL NO : 40**

**DATE : 14/09/22**

**1.OBJECTIVE**

WAP TO FIND THE ROOT OF ANY POLYNOMIAL EXPRESSION USING BISECTION METHOD CORRECT UPTO 3 DECIMAL PLACES

**2.METHOD /ALGORITHM**

1. Read x1, x2, e  
   \*Here x1 and x2 are initial guesses  
   e is the absolute error i.e. the desired degree of accuracy\*
2. Compute: f1 = f(x1) and f2 = f(x2)
3. If (f1\*f2) > 0, then display initial guesses are wrong
4. Otherwise, continue.
5. x = (x1 + x2)/2
6. If ( [ (x1 – x2)/x ] < e ), then display x   
   \* Here [ ] refers to the modulus sign. \*
7. Else, f = f(x)
8. If ((f\*f1) > 0), then x1 = x and f1 = f.
9. Else, x2 = x and f2 = f.

**3.PROGRAM**

#include <stdio.h>

#include <math.h>

float fun(float x)

{

return (pow(x, 3) - (4 \* x) - 9);

}

int main()

{

int m, n, x1, x2, x3;

float a, b;

printf("enter the range ");

scanf("%d %d", &m, &n);

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

{

if ((fun((float)i)) \* (fun((float)(i + 1))) < 0)

{

a = (float)i;

b = (float)(i + 1);

break;

}

}

printf("a is %f and b is %f\n", a, b);

int i = 1;

float x = (a + b) / 2;

do

{

printf("value at iteration x%d is %f\n ", i, x);

if (fun(a) \* fun(x) < 0)

b = x;

else if (fun(x) \* fun(b) < 0)

a = x;

x = (a + b) / 2;

i++;

} while (fabs(x - a) >= 0.0001 || fabs(x - b) >= 0.0001);

}

**4.OUTPUT :**

enter the range 0 5

a is 2.000000 and b is 3.000000

value at iteration x1 is 2.500000

value at iteration x2 is 2.750000

value at iteration x3 is 2.625000

value at iteration x4 is 2.687500

value at iteration x5 is 2.718750

value at iteration x6 is 2.703125

value at iteration x7 is 2.710938

value at iteration x8 is 2.707031

value at iteration x9 is 2.705078

value at iteration x10 is 2.706055

value at iteration x11 is 2.706543

value at iteration x12 is 2.706299

value at iteration x13 is 2.706421

**PROGRAM NO 3**

**NAME : PARAS BHATT**

**COURSE /BRACH : BTECH(CSE)**

**SEM/SEC : 5(E)**

**ROLL NO : 40**

**DATE : 14/09/22**

**1.OBJECTIVE**

WAP TO FIND THE ROOT OF ANY POLYNOMIAL EXPRESSION USING REGULAR FALSY METHOD CORRECT UPTO 3 DECIMAL PLACES

**2.METHOD /ALGORITHM**

* Read values of x0, x1 and e  
  \*Here x0 and x1 are the two initial guesses  
  e is the degree of accuracy or the absolute error i.e. the stopping criteria\*
* Computer function values f(x0) and f(x1)
* Check whether the product of f(x0) and f(x1) is negative or not. If it is positive take another initial guesses. If it is negative then goto step 5.
* Determine: x = [x0\*f(x1) – x1\*f(x0)] / (f(x1) – f(x0))
* Check whether the product of f(x1) and f(x) is negative or not.  
  If it is negative, then assign x0 = x;  
  If it is positive, assign x1 = x;
* Check whether the value of f(x) is greater than 0.00001 or not.  
  If yes, go to step 5.  
  If no, go to step 8.  
  \*Here the value 0.00001 is the desired degree of accuracy, and hence the stopping criteria.\*
* Display the root as x

**3.PROGRAM**

#include <stdio.h>

#include <math.h>

float function\_return(float x)

{

float res;

res = pow(x, 3) - (4 \* x) - 9;

return res;

}

int find\_points()

{

float res;

float res1;

for (int i = 0; i < 5; i++)

{

res = function\_return((float)i);

res1 = function\_return((float)(i + 1));

if (res1 < 0 && res > 0 || res1 > 0 && res < 0)

return (float)i;

}

return 0;

}

float truncate(float f, int p)

{

float temp = f \* pow(10, p);

int res = (int)(temp);

temp = (float)res / pow(10, p);

return temp;

}

int main()

{

float root, temp = 0.0, diff, mid, count;

float f\_i, s\_i;

int c = 1;

f\_i = find\_points();

s\_i = f\_i + 1;

do

{

root = function\_return(mid);

mid = (f\_i \* function\_return(s\_i) - s\_i \* function\_return(f\_i))/(function\_return(s\_i) - function\_return(f\_i));

if (function\_return(f\_i) \* function\_return(mid) < 0)

s\_i = mid;

else if (function\_return(mid) \* function\_return(s\_i) < 0)

f\_i = mid;

printf("%d root is : %f\n", c, mid);

diff = mid - temp;

temp = mid;

if (fabs(diff \* 1000) < 1)

{

printf("root is : %f\n", truncate(mid, 3));

break;

}

c++;

} while (1);

return 0;

}

**4.OUTPUT :**

enter the range 0 5

a is 2.000000 and b is 3.000000

value at iteration x1 is 2.500000

value at iteration x2 is 2.750000

value at iteration x3 is 2.625000

value at iteration x4 is 2.687500

value at iteration x5 is 2.718750

value at iteration x6 is 2.703125

value at iteration x7 is 2.710938

value at iteration x8 is 2.707031

value at iteration x9 is 2.705078

value at iteration x10 is 2.706055

value at iteration x11 is 2.706543

value at iteration x12 is 2.706299

value at iteration x13 is 2.706421