DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Year: 2rd Semester: 3rd

CBNST Lab-PCS-302

LAB MANUAL

Prepared By: HOD (CSE)

DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

INDEX

S.No	Practical's Name	Date	Remark
1	Write a program in "C" Language To deduce error envolved in polynomial equation.		
2	Write a program in "C" Language to find out the root of the Algebraic and Transcendental equations using Bisection Method		
3	Write a program in "C" Language to implement Newton's Forward and Backward Interpolation formula.		
4	Write a program in "C" Language to implement Gauss Forward and Backward interpolation formula.		
5	Write a program in "C" Language to implement Bessel's interpolation formula.		
6	Write a program in "C" Language to implement Sterling's Interpolation Formula.		
7	Write a program in "C" Language to implement Newton's Divided Difference formula.		
8	Write a program in "C" Language to implement lagrange's interpolation formula.		
9	Write a program in "C" Language to implement Least Square Method for curve fitting.		
10	Write a program in "C" Language to implement trapezoidal rule.		
11	Write a program in "C" Language to implement Simpson 3/8 rule.		

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 1**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to deduce error involved in polynomial equation.

Algorithm:

Step-1. Start of the program.

Step-2. Input the variable t_val, a_value.

Step-3. Calculate absolute error as

abs_err=|t_val-a_val|

Step-4. Calculate relative error as

rel_err=abs_err/t_val

Step-5. Calculate percentage relative error as

p_rel_err=rel_err*100

Step-6. PRINT abs_err, rel_err and p_rel_err

Step-7. STOP

Program:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
```

```
doubleabs_err, rel_err, p_rel_err, t_val, a_val;
printf("\n INPUT TRUE VALUE:");
scanf("%lf", &t_val);
printf("\n INPUT APPROXIMATE VALUE:");
scanf("%lf", &a_val);
abs_err=fabs(t_val-a_val);
rel_err=abs_err/t_val;
p_rel_err=rel_err*100;
printf("\nABSOLUTE ERROR= %lf", abs_err);
printf("\nRELATIVE ERROR= %lf", rel_err);
printf("\nPERCENTAGE RELATIVE ERROR= %lf", p_rel_err);
getch();
}
```

OUTCOME:

1) To deduce error involved in polynomial equation using C programming.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 2**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to find out the root of the Algebraic and Transcendental equations using Bisection Method.

Algorithm:

Step-1. Start of the program.

Step-2. Input the variable x1, x2 for the task.

Step-3. Check f(x1)*f(x2)<0

Step-4. If yes proceed

Step-5. If no exit and print error message

Step-6. Repeat 7-11 if condition not satisfied

Step-7. X0=(x1+x2)/2

Step-8. If f(x0)*f(x1)<0

Step-9. X2=x0

Step-10. Else

Step-11. X1=x0

Step-12. Condition:

Step-13.if | (x1-x2)/x1) | < maximum possible error or f(x0)=0

Step-14. Print output

Step-15. End of program.

Program:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#define EPS 0.00000005
#define F(x) (x)*log10(x)-1.2
void Bisect();
int count=1,n;
float root=1;
void main()
 {
clrscr();
printf("\n Solution by BISECTION method \n");
printf("\n Equation is ");
printf("\n\t\t\t x*log(x) - 1.2 = 0\n\n");
printf("Enter the number of iterations:");
scanf("%d",&n);
Bisect();
getch();
 }
void Bisect()
 {
float x0,x1,x2;
float f0,f1,f2;
inti=0;
```

```
for(x2=1;;x2++)
{
 f2=F(x2);
if (f2>0)
{
break;
}
 }
for(x1=x2-1;;x2--)
{
 f1=F(x1);
if(f1<0)
 {
break;
}
 }
printf("\t\t----------------------------------------");
{
 f2=F(x2);
if (f2>0)
{
break;
}
 }
for(x1=x2-1;;x2--)
{
 f1=F(x1);
```

```
if(f1<0)
 {
break;
}
 }
printf("\t\t----------------------------------------");
printf("\n\t\t----------------------------------------");
printf("\n\t\t\t Root = %7.4f",x0);
printf("\n\t\t\t Iterations = %d\n", count-1);
printf("\t\t----------------------------------------");
getch();
}
```

OUTCOME:

1) To find out the root of the Algebraic and Transcendental equations using Bisection Method using C programming.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 3**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implement Newton's Forward and Backward Interpolation formula.

Algorithm: Newton's Forward method of interpolation.

Step-1. Start of the program

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h=ax[1] – ax[0]

Step-6.fori=0; i<n-1; i++

Step-7.diff[i] [1]=ay[i + 1] – ay[i]

Step-8. End Loop i

Step-9.for j=2; j<=4; j++

Step-10.fori = 0; i<n – j; i++

Step-11.diff[i][j]=diff [i + 1] [j – 1]-diff [i][j – 1]

Step-12. End Loop i

Step-13. End Loop j

Step-14.i=0

Step-15. Repeat Step 16 until ax[i]<x

Step-16.i=i + 1

Step-17.i=i – 1;

Step-18. p=(x – ax [i])/h

Step-19. y1=p*diff[i – 1][1]

Step-20. y2=p*(p+1)*diff [i – 1][2]/2

Step-21. y3=(p+1)*p*(p-1)*diff[i –2][3]/6

Step-22. y4=(p+2)*(p+1)*p*(p – 1)*diff[i – 3][4]/24

Step-23. y=ay[i]+y1+y2+y3+y4

Step-24. Print output x, y

Step-25. End of program.

Program:

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
int n;
inti,j;
float ax[10];
float ay[10];
float x;
float y = 0;
float h;
float p;
float diff[20][20];
float y1,y2,y3,y4;
```

```
clrscr();
printf("\n Enter the number of terms - ");
scanf("%d",&n);
printf("Enter the value in the form of x - ");
for (i=0;i<n;i++)
{
printf("Enter the value of x%d - ",i+1);
scanf("%f",&ax[i]);
}
printf("\n Enter the value in the form of y - ");
for (i=0;i<n;i++)
 {
printf ("Enter the value of y%d - ", i+1);
scanf ("%f",&ay [i]);
}
printf("\nEnter the value of x for");
printf("\nwhich you want the value of y - ");
scanf("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
 }
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
 {
```

```
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
 }
 }
i=0;
do
 {
i++;
 }
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i-1][1];
y2=p*(p+1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+2)*(p+1)*p*(p-1)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\nwhen x=%6.4f, y=%6.8f ",x,y);
getch();
}
```

Algorithm: Newton's Backward method of interpolation.

Step-1. Start of the program.

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h=ax[1]-ax[0]

Step-6.fori=0; i<n–1; i++

Step-7.diff[i][1]=ay[i+1]–ay[i]

Step-8. End Loop i

Step-9.for j = 2; j < = 4; j + +

Step-10.fori=0; i<n–j; i++

Step-11.diff[i][j]=diff[i+1][j–1]–diff [i][j–1]

Step-12. End Loop i

Step-13. End Loop j

Step-14.i=0

Step-15. Repeat Step 16 until (!ax[i]<x)

Step-16.i=i+1

Step-17.x0=mx[i]

Step-18.sum=0

Step-19. y0=my[i]

Step-20.fun=1

Step-21. p=(x–x0)/h

Step-22.sum=y0

Step-23.for k=1; k<=4; k++

Step-24.fun=(fun*(p–(k–1)))/k

Step-25.sum=sum+fun*diff[i][k]

Step-26. End loop k

Step-27. Print Output x,sum

Step-28. End of Program

Program:

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

```
#include<process.h>
#include<string.h>
void main()
{
intn,i,j,k;
float mx[10],my[10],x,x0=0,y0,sum,h,fun,p,diff[20][20],y1,y2,y3,y4;
clrscr();
printf("\n enter the no. of terms - ");
scanf("%d",&n);
printf("\n enter the value in the form of x - ");
for(i=0;i<n;i++)
{
printf("\n enter the value of x%d- ",i+1);
scanf("%f",&mx[i]);
}
printf("\n enter the value in the form of y - ");
for(i=0;i<n;i++)
{
printf("\n\n enter the value of y%d- ",i+1);
scanf("%f",&my[i]);
}
printf("\n enter the value of x for");
printf("\nwhich you want the value of of y -");
scanf("%f",&x);h=mx[1]-mx[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=my[i+1]-my[i];
```

```
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
while(!mx[i]>x)
{
i++;
}
x0=mx[i];
sum=0;
y0=my[i];
fun=1;
p=(x-x0)/h;
sum=y0;
for(k=1;k<=4;k++)
{
fun=(fun*(p-(k-1))/k);
sum=sum+fun*diff[i][k];}
printf("\n when x=%6.4f,y=%6.8f",x,sum);
printf("\n press enter to exit");
getch();
}
```

OUTCOME:

1) To Newton's Forward and Backward Interpolation formula using C programming.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 4**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implement Gauss Forward and Backward interpolation formula.

Algorithm: Gauss Forward method of interpolation.

Step-1. Start of the program.

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h=ax[1]-ax[0]

Step-6.fori=0;i<n–1;i++

Step-7.diff[i][1]=ay[i+1]-ay[i]

Step-8. End Loop i

Step-9.for j=2;j<=4;j++

Step-10.fori=0;i<n–j;i++

Step-11.diff[i][j]=diff[i+1][j–1]–diff[i][j–1]

Step-12. End Loop i

Step-13. End Loop j

Step-14.i=0

Step-15. Repeat Step 16 until ax[i]<x

Step-16.i=i+1

Step-17.i=i–1;

Step-18. p=(x–ax[i])/h

Step-19. y1=p*diff[i][1]

Step-20. y2=p*(p–1)*diff[i–1][2]/2

Step-21. y3=(p+1)*p*(p-1)*diff[i–2][3]/6

Step-22. y4=(p+1)*p*(p–1)*(p–2)*diff[i–3][4]/24

Step-23. y=ay[i]+y1+y2+y3+y4

Step-24. Print Output x,y

Step-25.End of Program.

Program:

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
int n;
inti,j;
float ax[10];
float ay[10];
float x;
floatnr,dr;
float y=0; float h;
float p;
float diff[20][20];
float y1,y2,y3,y4;
```

```
clrscr();
printf(" Enter the number of terms - ");
scanf("%d",&n);
printf("\n Enter the value in the form of x - ");
for (i=0;i<n;i++)
{
printf(" Enter the value of x%d - ",i+1);
scanf("%f",&ax[i]);
}
printf(" Enter the value in the form of y - ");
for(i=0;i<n;i++)
{
printf("Enter the value of y%d - ",i+1);
scanf("%f",&ay[i]);
}
printf("\nEnter the value of x for - ");
printf("\nwhich you want the value of y - ");
scanf ("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
```

```
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
do {
i++;
}
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i][1];
y2=p*(p-1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+1)*p*(p-1)*(p-2)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\nwhen x=%6.4f,y=%6.8f ",x,y);
getch();
}
```

Algorithm: Gauss Backward method of interpolation.

Step-1. Start of the program.

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h=ax[1]-ax[0]

Step-6.fori=0;i<n-l;i++

Step-7.diff[i][1]=ay[i+1]-ay[i]

Step-8. End Loop i

Step-9.for j=2;j<=4;j++

Step-10.fori=0;i<n–j;i++

Step-11.diff[i][j]=diff[i+1][j–1]–diff[i][j–1]

Step-12. End Loop i

Step-13. End Loop j

Step-14.i=0

Step-15. Repeat Step 16 until ax[i]<x

Step-16.i=i+1

Step-17.i=i–1;

Step-18. p=(x–ax[i])/h

Step-19. y1=p*diff[i-1][1]

Step-20. y2=p*(p+1)*diff[i–1][2]/2

Step-21. y3=(p+1)*p*(p-1)*diff[i–2][3]/6

Step-22. y4=(p+2)*(p+1)*p*(p–1)*diff[i–3][4]/24

Step-23. y=ay[i]+y1+y2+y3+y4

Step-24. Print Output x,y

Step-25.End of Program.

Program:

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
```

```
int n;
inti,j; float ax[10];
float ay[10];
float x;
float y=0;
float h;
float p;
float diff[20][20];
float y1,y2,y3,y4;
clrscr();
printf("\n Enter the number of terms - ");
scanf("%d",&n);
printf("\n Enter the value in the form of x - ");
for (i=0;i<n;i++)
{
printf("\n\n Enter the value of x%d - ",i+1);
scanf("%f",&ax[i]);
}
printf("\n\n Enter the value in the form of y - ");
for(i=0;i<n;i++)
{
printf("\n Enter the value of y%d - ",i+1);
scanf("%f",&ay[i]);
}
printf("\nEnter the value of x for - ");
printf("\nwhich you want the value of y - ");
scanf("%f",&x);
```

```
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
do {
i++;
}
while (ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i-1][1];
y2=p*(p+1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+2)*(p+1)*p*(p-1)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\nwhen x=%6.1f,y=%6.4f ",x,y);
getch();
}
```

OUTCOME:

1) To implement Gauss Forward and Backward interpolation formula using C programming.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 5**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language toimplement Bessel's interpolation formula.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<conio.h>
void main()
{
int n; // no. of terms.
inti,j; // Loop variables
float ax[10]; // 'X' array limit 9
float ay[10]; // 'Y' array limit 9
float x; // User Query for what value of X
float y; // Calculated value for corresponding X.
float h; // Calc. Section
float p; // Calc. Section
float diff[20][20]; // to store Y
float y1,y2,y3,y4; // Formulae variables.
clrscr();
```

```
printf("\t\t !! BESSELS INTERPOLATION FORMULA!! ");
// Input section.
printf("\n\n Enter the no. of terms -> ");
scanf("%d",&n);
// Input Sequel for array X
printf("\n\n Enter the value in the form of x -> ");
// Input loop for X.
for(i=0;i<n;i++)
{
printf("\n Enter the value of x%d -> ",i+1);
scanf("%f",&ax[i]);
}
// Input sequel for array Y.
printf("\n\n Enter the value in the form of y -> ");
// Input loop for Y.
for(i=0;i<n;i++)
{
printf("\n Enter the value of y%d -> ",i+1);
scanf("%f",&ay[i]);
}
// Inputting the required value quarry
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<conio.h>
void main()
```

```
{
int n; // no. of terms.
inti,j; // Loop variables
float ax[10]; // 'X' array limit 9
float ay[10]; // 'Y' array limit 9
float x; // User Query for what value of X
float y; // Calculated value for coressponding X.
float h; // Calc. Section
float p; // Calc. Section
float diff[20][20]; // to store Y
float y1,y2,y3,y4; // Formulae variables.
clrscr();
printf("\t\t !! BESSELS INTERPOLATION FORMULA!! ");
// Input section.
printf("\n\n Enter the no. of terms -> ");
scanf("%d",&n);
// Input Sequel for array X
printf("\n\n Enter the value in the form of x -> ");
// Input loop for X.
for(i=0;i<n;i++)
{
printf("\n Enter the value of x%d -> ",i+1);
scanf("%f",&ax[i]);
}
// Input sequel for array Y.
printf("\n\n Enter the value in the form of y -> ");
// Input loop for Y.
```

```
for(i=0;i<n;i++)
{
printf("\n Enter the value of y%d -> ",i+1);
scanf("%f",&ay[i]);
}
// Inputting the required value quarry
printf("\n\n Enter the value of x for ");
printf("\n which u want the value of y -> ");
scanf("%f",&x);
// Calculation and processing section.
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
diff[i][1]=ay[i+1]-ay[i];
for(j=2;j<=4;j++)
for(i=0;i<n-j;i++)
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
i=0;
do
{
i++;
}
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i][1];
y2=p*(p-1)*(diff[i][2]+diff[i-1][2])/4;
y3=p*(p-1)*(p-0.5)*diff[i-1][3]/6;
```

```
y4=(p+1)*p*(p-1)*(p-2)*(diff[i-2][4]+diff[i-1][4])/48;
// Taking sum
y=ay[i]+y1+y2+y3+y4;
// Output Section
printf("\n When x = %6.2f , y = %6.8f",x,y);
// Invoke user watch halt function
printf("\n\n\n\t\t\t !! PRESS ENTER TO EXIT !! ");
getch();
}
```

OUTCOME:

1) To implementBessel's interpolation formulausing C programming.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 6**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implement Sterling's Interpolation Formula.

Algorithm:

Step-1. Start of the program.

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5. h = ax[1]-ax[0]

Step-6.fori = 1;i < n-1; i++

Step-7.diff [i][1] = ay[i + 1]-ay[i]

Step-8. End loop i

Step-9.for j = 2; j < = 4; j++

Step-10.fori = 0; i< n-j; i++

Step-11.diff[i][j] = diff[i + 1][j-1]-diff[i][j-1]

Step-12. End loop i

Step-13. End loop j

Step-14.i = 0

Step-15. Repeat step 16 until ax[i] < x

Step-16.i = i + 1

Step-17.i = i-1;

Step-18. p = (x-ax[i])/h

Step-19. y1= p*(diff[i][1] + diff[i-1][1])/2

Step-20. y2 = p*p*diff[i-1][2]/2

Step-21. y3 = p*(p*p-1)*(diff[i-1][3]+diff[i-2][3])/6

Step-22. y4 = p*p*(p*p-1)*diff[i-2][4]/24

Step-23. y = ay[i]+y1 + y2 + y3 + y4

Step-24. Print output

Step-25. End of program

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
void main()
{
int n;
inti,j;
float ax[10];
float ax[10];
float h;
float p;
float diff[20][20];
floatx,y;
float y1,y2,y3,y4;
clrscr();
printf("\n Enter the value of terms");
scanf("%d",%n);
printf(”\n Enter the values for x \n”);
```

```
for(i=0;i<n;i++)
{
printf("\n Enter the value for x%d-",i+1);
scanf("%f’,&ax[i]);
}
printf("\n Enter the values for y \n");
for(i=0;i<n;i++)
{
printf("\n Enter the value for y%d-",i+1);
scanf("%f",&ay[i]);
}
printf("\n Enter the value of x for");
printf("\n which you want the value of y");
scanf("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
```

```
do {
i++;
}
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*(diff[i][1]+diff[i-1][1])/2;
y2=p*p*diff[i-1][2]/2;
y3=p*(p*p-1)*(diff[i-1][3]+diff[i-2][3])/6;
y4=p*p*(p*p-1)*diff[i-2][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\n\n When x=%6.2f, y=%6.8f",x,y);
getch();
}
```

OUTCOME:

1) To implement Sterling's Interpolation Formula using C programming.

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 7**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implement Newton's Divided Difference formula.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
float x[10],y[10][10],sum,p,u,temp;
inti,n,j,k=0,f,m;
float fact(int);
clrscr();
printf("\nhow many record you will be enter: ");
scanf("%d",&n);
for(i=0; i<n; i++)
 {
printf("\n\nenter the value of x%d: ",i);
scanf("%f",&x[i]);
printf("\n\nenter the value of f(x%d): ",i);
scanf("%f",&y[k][i]);
 }
printf("\n\nEnter X for finding f(x): ");
scanf("%f",&p);
```

```
for(i=1;i<n;i++)
 {
 k=i;
for(j=0;j<n-i;j++)
 {
y[i][j]=(y[i-1][j+1]-y[i-1][j])/(x[k]-x[j]);
k++;
 }
 }
printf("\n_____\n");
printf("\n x(i)\t y(i)\t y1(i) y2(i) y3(i) y4(i)");
printf("\n_____\n");
for(i=0;i<n;i++)
 {
printf("\n %.3f",x[i]);
for(j=0;j<n-i;j++)
 {
printf(" ");
printf(" %.3f",y[j][i]);
 }
printf("\n");
}
i=0;
do
 {
if(x[i]<p && p<x[i+1])
 k=1;
```

```
else
i++;
}while(k != 1);
 f=i;
sum=0;
for(i=0;i<n-1;i++)
 {
 k=f;
temp=1;
for(j=0;j<i;j++)
 {
temp = temp * (p - x[k]);
k++;
 }
sum = sum + temp*(y[i][f]);
 }
printf("\n\n f(%.2f) = %f ",p,sum);
getch();
}
```

OUTCOME:

1) To implement Newton's Divided Difference formulausing C programming.

LAB MANUAL			
	Course Name :CBNST Lab	**EXPERIMENT NO. 8**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implementlagrange's interpolation formula.

Algorithm:

Step-1. Start of the program

Step-2. Input number of terms n

Step-3. Input the array ax

Step-4. Input the array ay

Step-5.fori=0; i<n; i++

Step-6.nr=1

Step-7.dr=1

Step-8.for j=0; j<n; j++

Step-9.if j !=i

a. nr=nr*(x-ax[j])

Step-10.b.dr*(ax[i]-ax[j])

Step-11. End Loop j

Step-12. y+=(nr/dr)∗ay[i]

Step-13. End Loop i

Step-14. Print Output x, y

Step-15. End of Program

Program:

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

```
#define MAX 10
void main()
{
float x[MAX],y[MAX],k=0,z,nr,dr;
inti,j,m;
//clrscr();
printf("\n enter the range ");
scanf("%d",&m);
printf("\n enter the x value ");
for(i=0;i<m;i++)
scanf("%f",&x[i]);
printf("\n enter the y value ");
for(i=0;i<m;i++)
scanf("%f",&y[i]);
printf("\n enter value OF Z to be calculated ");
scanf("%f",&z);
for(i=0;i<m;i++)
{ nr=1;dr=1;
for(j=0;j<m;j++)
{
if (j!=i)
{
nr=nr*(z-x[j]);
dr=dr*(x[i]-x[j]);
}
}
k=k+((nr/dr)*y[i]);
```

```
}
printf("\n final result=%f\n",k);
getch();}
```

OUTCOME:

2) To implementlagrange's interpolation using C programming.

LAB MANUAL

	Course Name :CBNST Lab	**EXPERIMENT NO. 9**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language toimplement Least Square Method for curve fitting.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
float x[10],y[10],a[10][10];
inti,j,k,n,itr;
printf("\n ENTER THE SIZE OF MATRIX n:");
scanf("%d",&n);
printf("\n ENTER MATRIX ELEMENTS AND RHS:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n+1;j++)
scanf("%f",&a[i][j]);
}
for(i=1;i<=n;i++)
{
x[i]=0.0;
y[j]=0.0;
}
```

```
itr=0.0;
top:
itr=itr+1;
for(i=1;i<=n;i++)
{
x[i]=a[i][n+1];
for(j=1;j<=n;j++)
{
if(i==j)
continue;
else
x[i]=x[i]-a[i][j]*x[j];
}
x[i]=x[i]/a[i][j];
}
for(k=1;k<=n;k++)
if(fabs(x[k]-y[k])>0.0001)
{
printf("\n ITERATION=%d}",itr);
for(i=1;i<=n;i++)
{
y[i]=x[i];
printf("\n x(%d)=%f",i,x[i]);
}
goto top;
}
else
```

```
continue;
return;
}
```

OUTCOME:

1) To implement Least Square Method for curve fitting using C programming.

LAB MANUAL			
	Course Name :CBNST Lab	**EXPERIMENT NO. 10**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implement trapezoidal rule.

Algorithm:

Step-1. Start of the program.

Step-2. Input Lower limit a

Step-3. Input Upper Limit b

Step-4. Input number of sub intervals n

Step-5. h=(b-a)/n

Step-6.sum=0

Step-7.sum=fun(a)+fun(b)

Step-8.fori=1; i<n; i++

Step-9.sum +=2*fun(a+i)

Step-10. End Loop i

Step-11.result =sum*h/2;

Step-12. Print Output result

Step-13. End of Program

Step-14. Start of Section fun

Step-15.temp = 1/(1+(x*x))

Step-16. Return temp

Step-17. End of Section fun

Program:

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

```
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
float fun(float);
void main()
{
float result=1;
floata,b;
floath,sum;
inti,j;
int n;
clrscr();
printf("\n\n Enter the range - ");
printf("\n\n Lower Limit a - ");
scanf("%f" ,&a);
printf("\n\n Upper Limit b - ");
scanf("%f" ,&b);
printf("\n\n Enter number of subintervals - ");
scanf("%d" ,&n);
h=(b-a)/n;
sum=0;
sum=fun(a)+fun(b);
for(i=1;i<n;i++)
{
sum+=2*fun(a+i);
}
```

```
result=sum*h/2;
printf("n\n\n\n Value of the integral is %6.4f\t",result);
printf("\n\n\n Press Enter to Exit");
getch();
}
float fun(float x)
{
float temp;
temp = 1/(1+(x*x));
return temp;
}
```

OUTCOME:

1) To implement trapezoidal ruleusing C programming.

LAB MANUAL			
	Course Name :CBNST Lab	**EXPERIMENT NO. 11**	
	Course Code : PCS 302 **Faculty : Mr. Raman Raghav**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program in "C" Language to implement Simpson 3/8 rule.

Algorithm:

Step-1. Start of the program.

Step-2. Input Lower limit a

Step-3. Input Upper limit b

Step-4. Input number of subintervals n

Step-5. h=(b–a)/n

Step-6.sum=0

Step-7.sum=fun(a)+4*fun(a+h)+fun(b)

Step-8.fori=3; i<n; i + = 2

Step-9.sum + = 2*fun(a+(i – 1)*h) + 4*fun(a+i*h)

Step-10. End of loop i

Step-11.result=sum*h/3

Step-12. Print Output result

Step-13. End of Program

Step-14. Start of Section fun

Step-15.temp = 1/(1+(x*x))

Step-16. Return temp

Step-17. End of Section fun

Program:

#include<stdio.h>

```
#include<conio.h>
#include<math.h>
#include<process.h>
#include<string.h>
float fun(float);
void main()
{
float result=1;
floata,b;
floatsum,h;
inti,j,n;
clrscr();
printf("\n Enter the range - ");
printf("\n Lower Limit a - ");
scanf("%f",&a)
;printf("\n Upper limit b - ");
scanf("%f",&b);
printf("\n\n Enter number of subintervals - ");
scanf("%d",&n);
h=(b-a)/n;
sum=0;
sum=fun(a)+4*fun(a+h)+fun(b);
for(i=3;i<n;i+=2)
{
sum+=2*fun(a+(i-1)*h)+4*fun(a+i*h);
}
result=sum*h/3;
```

```
printf("\n\nValue of integral is %6.4f\t",result);
getch();}
float fun(float x)
{
float temp;
temp=1/(1+(x*x));
return temp;
}
```

OUTCOME:

1) To implement Simpson 3/8 rule using C programming.