Assignment Number

Problem Statement

Program in C to approximate a value of f(x) for a given value of x using Newton's Forward Interpolation.

Theory

Interpolation is the technique of estimating the value of a function for any intermediate value of the independent variable, while the process of computing the value of the function outside the given range is called **extrapolation**.

Forward Differences : The differences y1 – y0, y2 – y1, y3 – y2, ……, yn – yn–1 when denoted by dy0, dy1, dy2, ……, dyn–1 are respectively, called the first forward differences. Thus the first forward difference are:

$$\Delta Y_r = Y_{r+1} - Y_r$$

Forward difference table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0					
		Δy_0				
x_1 $(= x_0 + h)$	y_1		$\Delta^2 y_0$			
		Δy_1		$\Delta^3 y_0$		
x_2 $(= x_0 + 2h)$	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$	
		Δy_2		$\Delta^3 y_1$		$\Delta^5 y_0$
x_3 $= (x_0 + 3h)$	y_3		$\Delta^2 y_2$		$\Delta^4 y_1$	
		Δy_3		$\Delta^3 y_2$		
x_4 $= (x_0 + 4h)$	y_4		$\Delta^2 y_3$			
		Δy_4				
x_5 $= (x_0 + 5h)$	y_5					

Newton's_Forward_Interpolation_Formula:

$$f(a+hu)=f(a)+u\Delta f(a)+\frac{u(u-1)}{2!}\Delta^2 f(a)+\frac{u(u-1)(u-2)..(u-n+1)}{u!}\Delta^n f(a)$$

This formula is particularly useful for interpolating the values of f(x) near the beginning of the set of values given. h is called the interval of difference and **u = (x – a) / h**, Here a is first term.

Algorithm

Input :

1. Several values of X_i and $y_i = f(X_i)$
2. The value to approximate f(x) for, say x
3. The width of each interval, say h

Output : The value of f(x) at given point

Steps :

Step 1 : At first, several values of X_i and the corresponding $f(X_i)$ is taken as input

Step 2 : Input x, for which f(x) is to be calculated

Step 3 : Input h

Step 4 : Prepare difference table using

$$\Delta^n f(x_i)=\Delta^{n-1} f(x_{i+1})-\Delta^{n-1} f(x_i)$$

Step 5 : Calculate the value of k by using
k = (x – x_0) / h ; where x is the predicate value, x_0 is the initial value, and h is the interval width

Step 6 : Substitute the value in the formula

$$K(x)=\prod_{r=0}^{i}\frac{(K-r)}{(r+1)}$$

Step 7 : $f(x)=\sum_{i=0}^{n} K(x)\Delta^i f(x_n)$

Step 8 : End

Source Code

```
#include<stdio.h>
int main()
{
	float x[10],y[10][10],sum,p,u,temp;
	int i,n,j,k=0,f,m;
	float fact(int);
	printf("\nHow many records you want to enter:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter the value of x%d: ",i);
		scanf("%f",&x[i]);
		printf("\nEnter the value of f(x%d):",i);
		scanf("%f",&y[k][i]);
	}
	printf("\nEnter X for finding f(x):");
	scanf("%f",&p);
	for(i=1;i<n;i++)
	{
		for(j=0;j<n-i;j++)
		{
			y[i][j]=y[i-1][j+1] - y[i-1][j];
		}
	}
	printf("\nThe Table is:\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;
        u=(p-x[f])/(x[f+1]-x[f]);
        printf("\n\n u= %.3f",u);
        n-n-i+1;
        sum=0;
        for(i=0;i<n-1;i++)
        {
                temp=1;
                for(j=0;j<i;j++)
                {
                        temp= temp * (u-j);
                }
                m=fact(i);
                sum= sum +temp * (y[i][f]/m);
        }
        printf("\n\n f(%.2f) = %f",p,sum);
}
float fact(int a)
{
        float fac = 1;
        if(a==0)
                return 1;
        else
                fac = a * fact(a-1);
        return (fac);
}
```

Input and Output

Set 1:

How many records you want to enter:5

Enter the value of x0: 0

Enter the value of f(x0):-1

Enter the value of x1: 1

Enter the value of f(x1):0

Enter the value of x2: 2

Enter the value of f(x2):7

Enter the value of x3: 3

Enter the value of f(x3):26

Enter the value of x4: 4

Enter the value of f(x4):63

Enter X for finding f(x):1.5

The Table is:

0.000	-1.000	1.000	6.000	6.000	0.000
1.000	0.000	7.000	12.000	6.000	
2.000	7.000	19.000	18.000		

3.000   26.000   37.000

4.000   63.000

u= 0.500

f(1.50) = 2.375000

Discussion

1. We can see in our program we used array whose size is to be predefined. It will be convenient if some dynamic memory would be alloted for our required values of x and f(x) to be stored in. This will hault the problem of wastage of memory/scarcity of memory if array size is predefined.
2. Newton’s forward interpolation only gives precise results for values near the beginning of the difference table.
3. Newton’s forward interpolatioon method is not general enough to be applied to all possible scenarios.