// 3.1. Derivative using Newton's Divided Differences Table

/\*

#include <stdio.h>

#include <math.h>

#define MAX 15

int main()

{

int i, j, k, n;

float x[MAX], f[MAX], a[MAX], d[MAX][MAX], xp, dif, sum, p;

char q;

printf("Newton's Divided Difference : ");

printf("\n Input number of data points:");

scanf("%d", &n);

printf("\nInput values of x and f(x) one set on each line:\n");

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

scanf("%f%f", &x[i], &f[i]);

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

d[i][1] = f[i];

for (j = 2; j <= n; j++)

for (i = 1; i <= n - j + 1; i++)

d[i][j] = (d[i + 1][j - 1] - d[i][j - 1]) / (x[i + j - 1] - x[i]);

for (j = 1; j <= n; j++)

a[j] = d[1][j];

printf("\n Input xp where derivative is required: ");

scanf("%f", &xp);

dif = a[2];

for (k = 3; k <= n; k++)

{

sum = 0.0;

for (i = 1; i <= k - 1; i++)

{

p = 1.0;

for (j = 1; j <= k - 1; j++)

{

if (i != j)

p = p \* (xp - x[j]);

}

sum = sum + p;

}

dif = dif + a[k] \* sum;

}

printf("\n Derivative at x=%f is %f.", xp, dif);

return 0;

}

\*/

// central difference formula

/\*

#include <stdio.h>

#include <math.h>

float f(float x)

{

return exp(x) \* sqrt(sin(x) + log(x));

}

int main()

{

float a, h, fd1, fd2;

char q;

printf("Central Difference ");

printf("\nEnter the point at which derivatives are required: ");

scanf("%f", &a);

printf("\nEnter the value of h: ");

scanf("%f", &h);

fd1 = (f(a + h) - f(a - h)) / (2 \* h);

fd2 = (f(a + h) - 2 \* f(a) + f(a - h)) / (h \* h);

printf("\nTHe first and second derivative at x=%f are %f and %f respectively", a, fd1, fd2);

return 0;

}

\*/

// Composite Trapezoidal Rule

/\*

#include <stdio.h>

#include <math.h>

float f(float x)

{

return exp(-x \* x);

}

int main()

{

int n, i;

float a, b, h, sum = 0.0, ict;

printf("Composite Trapezoidal Rule");

printf("\n Give lower & upper limit of integration a,b:");

scanf("%f %f", &a, &b);

printf("\n Give the segment width h:");

scanf("%f", &h);

n = (b - a) / h;

for (i = 1; i <= n - 1; i++)

sum = sum + 2 \* f(a + i \* h);

sum = sum + f(a) + f(b);

ict = sum \* h / 2;

printf("\n Integration between %f and %f when h = %f is %f.", a, b, h, ict);

return 0;

}

\*/

// Composite simpson 1/3 rule

/\*

#include <stdio.h>

#include <math.h>

float f(float x)

{

return (x \* x) \* (cos(x));

}

int main()

{

int n, m, i;

float a, b, h, sum = 0.0, ics, x;

printf("Composite Simpson's 1/3 ");

printf("\n Give lower & upper limit of integration a,b: ");

scanf("%f%f", &a, &b);

do

{

printf("\n Give number of segments n (Even number):");

scanf("%d", &n);

} while (n % 2 != 0);

h = (b - a) / n;

m = n / 2;

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

{

x = a + (2 \* i - 1) \* h;

sum = sum + 4 \* f(x);

if (i != m)

sum = sum + 2 \* f(x + h);

}

sum = sum + f(a) + f(b);

ics = sum \* h / 3.0;

printf("\n Integration between %f and %f when h=%f is %f.", a, b, h, ics);

return 0;

}

\*/

// composite Simpson's 3/8 rule

/\*

#include <stdio.h>

#include <math.h>

float f(float x)

{

return exp(-x \* x);

}

int main()

{

int n, m, i;

float a, b, h, sum = 0.0, ics, x;

printf("Composite Simpson's 3/8 ");

printf("\n Give lower & upper limits a,b :");

scanf("%f %f", &a, &b);

do

{

printf("\n Give number of segments n(divisible by 3):");

scanf("%d", &n);

} while (n % 3 != 0);

h = (b - a) / n;

m = n / 3;

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

{

x = a + (3 \* i - 2) \* h;

sum = sum + 3 \* (f(x) + f(x + h));

if (i != m)

sum = sum + 2 \* f(x + 2 \* h);

}

sum = sum + f(a) + f(b);

ics = sum \* 3 \* h / 8.0;

printf("\nIntegration between %f and %f when h = %f is %f.", a, b, h, ics);

return 0;

}

\*/