Assignment 5

Write a program in C to solve the following differential equation by Runge-Kutta 2^{nd} Order method.

$\frac{dy}{dx} + xy = 0, y(0) = 1\ from\ x = 0\ to\ x = 0.25$

Algorithm

Define f(x, y) as the given differential equation
Read x_0, y_0, h, n

For i from 0 to n-1 do:
    x = x_0 + (i * h)
    k_1 = h * f(x_0, y_0)
    k_2 = h * f(x_0 + h, y_0 + k_1)
    y_1 = y_0 + 0.5 * (k_1 + k_2)
    Print x, y
    x_0 = x
    y_0 = y_1
End for

End.

Source Code: RungeKuttaSecondOrder.c

```
#include <stdio.h>

// A sample differential equation
float dydx(float x, float y) { return -(x*y); }

// Finds value of y for a given x
// using step size h
// and initial value y0 at x0.
float rungeKutta(float x0, float y0, float x, float h)
{
	// Count number of iterations
	// using step size or
	// step height h
	int n = (int)((x - x0) / h);

	float k1, k2;

	// Iterate for number of iterations
	float y = y0;
	for (int i = 1; i <= n; i++) {
		// Apply Runge Kutta Formulas
		// to find next value of y
		k1 = h * dydx(x0, y);
		k2 = h * dydx(x0 + 0.5 * h, y + 0.5 * k1);

		// Update next value of y
		y = y + (1.0 / 6.0) * (k1 + 2 * k2);

		// Update next value of x
```

```
            x0 = x0 + h;
        }
        return y;
}

// Driver Code
int main()
{
        float x0, y0, x, h;
        printf("Enter the value of x0.\n");
        scanf("%f", &x0);
        printf("Enter the value of y0.\n");
        scanf("%f", &y0);
        printf("Enter the value of x.\n");
        scanf("%f", &x);
        printf("Enter the value of h.\n");
        scanf("%f", &h);
        printf("\nThe value of y at x is : %f\n",  rungeKutta(x0, y0, x, h));
        return 0;
}
```

Output

```
Enter the value of x0.
0
Enter the value of y0.
1
Enter the value of x.
0.25
Enter the value of h.
0.25

The value of y at x is : 0.989583
```

Assignment 6

Write a program in C to solve the following differential equation by Runge-Kutta 4^{th} Order method.

$\frac{dy}{dx} + xy = 0, y(0) = 1\ from\ x = 0\ to\ x = 0.25$

Algorithm

1. Define $f(x, y) = xy$
2. Read x_0, y_0, h, n
3. For I = 0 to $n - 1$ do
4. $x_{i+1} = x_i + h$
5. $d_1 = hf(x_i, y_i)$
6. $d_2 = hf(x_i + \frac{h}{2}, y_i + \frac{1}{2} d_1)$
7. $d_3 = hf(x_i + \frac{h}{2}, y_i + \frac{1}{2} d_2)$
8. $d_4 = hf(x_i + h, y_i + d_3)$
9. $y_{i+1} = \frac{1}{6}(d_1 + 2d_2 + 2d_3 + d_4)$
10. Print x_{i+1}, y_{i+1}
11. Next i
12. End.

Source Code: RungeKuttaFourthOrder.c

```
#include<stdio.h>

// A sample differential equation "dy/dx = (x - y)/2"
float dydx(float x, float y)
{
	return -(x*y);
}

// Finds value of y for a given x using step size h
// and initial value y0 at x0.
float rungeKutta(float x0, float y0, float x, float h)
{
	// Count number of iterations using step size or
	// step height h
	int n = (int)((x - x0) / h);

	float k1, k2, k3, k4, k5;

	// Iterate for number of iterations
	float y = y0;
	for (int i=1; i<=n; i++)
	{
		// Apply Runge Kutta Formulas to find
		// next value of y
		k1 = h*dydx(x0, y);
		k2 = h*dydx(x0 + 0.5*h, y + 0.5*k1);
		k3 = h*dydx(x0 + 0.5*h, y + 0.5*k2);
		k4 = h*dydx(x0 + h, y + k3);

		// Update next value of y
```

```
		y = y + (1.0/6.0)*(k1 + 2*k2 + 2*k3 + k4);;

		// Update next value of x
		x0 = x0 + h;
	}

	return y;
}

// Driver Code
int main()
{
	float x0, y0, x, h;
	printf("Enter the value of x0.\n");
	scanf("%f", &x0);
	printf("Enter the value of y0.\n");
	scanf("%f", &y0);
	printf("Enter the value of x.\n");
	scanf("%f", &x);
	printf("Enter the value of h.\n");
	scanf("%f", &h);

	printf("\nThe value of y at x is : %f\n",  rungeKutta(x0, y0, x, h));
	return 0;
}
```

Output

```
Enter the value of x0.
0
Enter the value of y0.
1
Enter the value of x.
0.25
Enter the value of h.
0.25

The value of y at x is : 0.969233
```

Assignment 7

Write a program in C to solve the following integration by Trapezoidal method.

$$\int_0^4 xe^{2x}\,dx$$

Algorithm

```
Define f(x) as the integrand function

Define trapezoidal(n, a, b) as the function to perform Trapezoidal Rule integration
    Calculate h as (b - a) / n
    Initialize sum as 0
    Initialize result as 0

    For i from 1 to n-1 do:
        Evaluate f(a + i*h) and add it to sum

    Calculate result as (h/2) * (f(a) + f(b) + 2 * sum)
    Print "The result is:", result

Define main() as the starting point of the program
    Declare variables: n, lower, upper

    Print "Enter the number of intervals."
    Read n

    Print "Enter the lower limit."
    Read lower

    Print "Enter the upper limit."
    Read upper

    Call trapezoidal(n, lower, upper)
    Return 0
```

Source Code: Trapezoidal.c

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

double f(double x) {
    return x * exp(2*x);
}

void trapezoidal(int n, double a, double b) {
    double h = (b-a)/n;
    double sum = 0, result;

    for(int i = 1; i < n; i++) {
        sum += f(a+i*h);
    }

    result = (h/2) * (f(a) + f(b) + 2 * sum);
```

```
    printf("\n The result is: %f\n", result);
}
int main() {
    int n;
    double lower, upper;
    printf("Enter the number of interval.\n");
    scanf("%d", &n);

    printf("Enter the lower limit.\n");
    scanf("%lf", &lower);

    printf("Enter the upper limit.\n");
    scanf("%lf", &upper);

    trapezoidal(n, lower, upper);
    return 0;
}
```

Output

```
Enter the number of interval.
300
Enter the lower limit.
0
Enter the upper limit.
4

 The result is: 5217.323918
```

Assignment 8

Write a program in C to solve the following integration by Simpson's $\frac{1}{3}rd$ rule.

$$\int_0^4 xe^{2x}\,dx$$

Algorithm

1.Select a value for n (n must be even), which is the number of parts the interval is divided into.

2.Calculate the width, h = (b-a)/n

3.Calculate the values of x0 to xn as x0 = a, x1 = x0 + h, ...xn-1 = xn-2 + h, xn = b.

Consider y = f(x). Now find the values of y(y0 to yn) for the corresponding x(x0 to xn) values.

4.Substitute all the above found values in the Simpson's Rule Formula to calculate the integral value.

Approximate value of the integral can be given by Simpson's Rule:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f_0 + f_n + 4 * \sum_{i=1,3,5}^{n-1} f_i + 2 * \sum_{i=2,4,6}^{n-2} f_i\right)$$

Source Code: SimpsonOneThird.c

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

double f(double x) {
  return x * exp(2*x);
}

void simpson(int n, double a, double b) {
  double h;
  double sum1 = 0, sum2 = 0, result;
  if(n%2==0) {
    h = (b-a)/n;
    for(int i = 1; i < n; i++) {
      if(i%2==0) {
        sum1 += f(a + i*h);
      } else {
        sum2 += f(a+i*h);
      }
    }

    result = (h/3) * (f(a) + f(b) + 2 * sum1 + 4 * sum2);

    printf("\n The result is: %f\n", result);

  } else {
    printf("\nNo. of intervals should be even.\n");
  }
}
int main() {
  int n;
```

```
    double lower, upper;
    printf("Enter the number of intervals.\n");
    scanf("%d", &n);

    printf("Enter the lower limit.\n");
    scanf("%lf", &lower);

    printf("Enter the upper limit.\n");
    scanf("%lf", &upper);

    simpson(n, lower, upper);

    return 0;
}
```

Output

```
Enter the number of intervals.
50
Enter the lower limit.
0
Enter the upper limit.
4

 The result is: 5216.956214
```