Tutorial 1 (Week 2)
Basic C Programming and Control Flow

Q1 (linearSystem)

Write a C program that computes the solutions for x and y in the linear system of equations:

$$a_1x + b_1y = c_1$$
$$a_2x + b_2y = c_2$$

The solutions for x and y are given by:

$$x = \frac{b_2c_1 - b_1c_2}{a_1b_2 - a_2b_1} \qquad \text{and} \qquad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

Test Case 1:

Enter the values for a1, b1, c1, a2, b2, c2:

1 1 1 5 7 9

x = -1.00 and y = 2.00

Test Case 2:

Enter the values for a1, b1, c1, a2, b2, c2:

1 1 2 2 3 3

x = 3.00 and y = -1.00

Q1 (linearSystem)

```
#include <stdio.h>
#include <math.h>
int main()
{
  float a1,b1,c1,a2,b2,c2;
  float x,y;

  printf("Enter the values for a1, b1, c1, a2, b2, c2: \n");
  scanf("%f %f %f %f %f %f", &a1, &b1, &c1, &a2, &b2, &c2);
  if (fabs(a1*b2 - a2*b1) >=  0.0001)
  {
    x = (b2*c1 - b1*c2) / (a1*b2 - a2*b1);
    y = (a1*c2 - a2*c1) / (a1*b2 - a2*b1);
    printf("x = %.2f and y = %.2f\n", x, y);
  }
  else
    printf("Unable to compute because the denominator is zero!");
  return 0;
}
```

Note:

- **Take note of floating point value 0, which can be represented as very small value.**

Q2 (countChars)

Write a C program that reads in character by character from an input source, until '#' is entered. The output of the program is the number of English letters and the number of digits that appear in the input.

Test Case 1:

Enter your characters (# to end):

happy 34567 fans#

The number of digits: 5

The number of letters: 9

Test Case 2:

Enter your characters (# to end):

1a2b3c#

The number of digits: 3

The number of letters: 3

Q2 (countChars)

```
#include <stdio.h>
int main()
{
  int ccount = 0, dcount = 0;
  char ch;
  printf("Enter your characters (# to end): \n");
  scanf("%c",&ch);
  while ( ch != '#') {
    if (ch >= '0' && ch <= '9')
      dcount++;
    else if ((ch >= 'A' && ch <= 'Z')  ||  (ch >= 'a' && ch <= 'z'))
      ccount++;
    scanf("%c",&ch);
  }
  printf("The number of digits: %d\n", dcount);
  printf("The number of letters: %d\n", ccount);
  return 0;
}
```

Q3 (printPattern)

(**printPattern**) Write a C program that accepts a positive number *height* between 1 and 10 as its parameter value, and prints a triangular pattern according to *height*. A sample input and output session when the program is called is given below.

For example, *pattern(7)* will print the pattern as shown. Note that only 1, 2 and 3 are used to generate the patterns.

Sample input and output session:

```
Enter the height:
7
Pattern:
1
22
333
1111
22222
333333
1111111
```

Q3 (printPattern)

```
#include <stdio.h>
int main()
{
    int row, col, height;
    int num = 0;
    printf("Enter the height: \n");
    scanf("%d", &height);
    printf("Pattern: \n");
    for (row = 0; row < height; row++)
    {

        for (col=0; col<row+1; col++) //print numbers
            printf("%d",num+1);
        num = (num + 1) % 3; // print up to number 3
        printf("\n");
    }
    return 0;
}
```

Sample input and output:

Enter the height:
7
Pattern:
1
22
333
1111
22222
333333
1111111

Note:

- **2-dimensional – row and column – we should use nested loop for the processing.**
- **Determine the number of rows via height.**
- **For each row, you need to print the number of times the number to be printed.**
- **When printing the number, you also need to use the modulus operator in order to limit the number to be printed. You may also use the if statement. For example:**

  num =num+1; if (num==3) num=0;

Q4 (computeSeries)

Write a C program that computes the value of e^x according to the following formula:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots + \frac{x^{10}}{10!}$$

Test Case 1:

Enter x:

0.9

Result = 2.46

Test Case 2:

Enter x:

0

Result = 1.00

Test Case 3:

Enter x:

-0.9

Result = 0.41

Q4 (computeSeries)

```
#include <stdio.h>
int main()
{
   int n, denominator = 1;
   float x, result = 1.0, numerator = 1.0;

   printf("Please enter the value of x: \n");
   scanf("%f", &x);
   for (n = 1; n <= 10; n++)
   {
      denominator *= n;
      numerator *= x;
      result += numerator/denominator;
   }
   printf("Result = %.2f\n",  result);
   return 0;
}
```

Please enter the value of x:

1

Result = 2.72

numerator

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots + \frac{x^{10}}{10!}$$

denominator

n = 1 2 3 10

Initial values: result=1.0; numerator=1.0; denominator=1;

- **When n=1, result = result+num/den = 1 + (1*x)/(1*n) = 1 + x/1!**
- **When n=2, result = 1 + x/1! + num/den = 1+x/1! + (1*x)*x/(1!*2) = 1+x/1! + x^2/2!**
- **When n=3, result = (1+x/1! + x^2/2!) + (x^2*x)/(2!*3) = 1+x/1! + x^2/2! + x^3/3!**
- **When n=4, result = 1+x/1! + x^2/2! + x^3/3! + x^4/4!**
- **Etc.**