CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 2, 3
Lab instructor: Marufa Ferdausi
Lab 06(a) – Spring 2018

Strings:

Task – 1
(*Student major and status*) Write a program that prompts the user to enter two characters and displays the major and status represented in the characters. The first character indicates the major and the second is number character 1, 2, 3, 4, which indicates whether a student is a freshman, sophomore, junior, or senior. Suppose the following chracters are used to denote the majors:

M: Mathematics
C: Computer Science
I: Information Technology

Here is a sample run:

```
Enter two characters: M1
Mathematics Freshman
```

```
Enter two characters: C3
Computer Science Junior
```

```
Enter two characters: T3
Invalid input
```

Task – 2
(*Check SSN*) Write a program that prompts the user to enter a Social Security number in the format DDD-DD-DDDD, where D is a digit. Your program should check whether the input is valid. Here are sample runs:

```
Enter a SSN: 232-23-5435
232-23-5435 is a valid social security number
```

```
Enter a SSN: 23-23-5435
23-23-5435 is an invalid social security number
```

Task – 3
(*Order three cities*) Write a program that prompts the user to enter three cities and displays them in ascending order. Here is a sample run:

```
Enter the first city: Chicago
Enter the second city: Los Angeles
Enter the third city: Atlanta
The three cities in alphabetical order are Atlanta Chicago Los Angeles
```

Methods:

Task – 4
(Check password) Some websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rules are as follows:
■ A password must have at least eight characters.
■ A password consists of only letters and digits.
■ A password must contain at least two digits.
Write a program that prompts the user to enter a password and displays **Valid Password** if the rules are followed or **Invalid Password** otherwise.

Task – 5
(*Count the letters in a string*) Write a method that counts the number of letters in a string using the following header:

public static int countLetters(String s)

Write a test program that prompts the user to enter a string and displays the number of letters in the string.

Task – 6
(*Math: approximate the square root*) There are several techniques for implementing the **sqrt** method in the **Math** class. One such technique is known as the *Babylonian method.* It approximates the square root of a number, **n**, by repeatedly performing a calculation using the following formula:

nextGuess = (lastGuess + n / lastGuess) / **2**

When **nextGuess** and **lastGuess** are almost identical, **nextGuess** is the approximated square root. The initial guess can be any positive value (e.g., **1**). This value will be the starting value for **lastGuess**. If the difference between **nextGuess** and **lastGuess** is less than a very small number, such as **0.0001**, you can claim that **nextGuess** is the approximated square root of **n**. If not, **next-Guess** becomes **lastGuess** and the approximation process continues. Implement the following method that returns the square root of **n**.

public static double sqrt(**long** n)

Single-Dimensional Arrays:

Task – 7
(*Merge two sorted lists*) Write the following method that merges two sorted lists into a new sorted list.

public static int[] merge(**int**[] list1, **int**[] list2)

Implement the method in a way that takes at most **list1.length + list2. length** comparisons. Write a test program that prompts the user to enter two sorted lists and displays the merged list. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list1: 5 1 5 16 61 111
Enter list2: 4 2 4 5 6
The merged list is 1 2 4 5 5 6 16 61 111
```

Task – 8
(*Partition of a list*) Write the following method that partitions the list using the first element, called a *pivot*.

public static int partition(**int**[] list)

After the partition, the elements in the list are rearranged so that all the elements before the pivot are less than or equal to the pivot and the elements after the pivot are greater than the pivot. The method returns the index where the pivot is located in the new list. For example, suppose the list is {5, 2, 9, 3, 6, 8}. After the partition, the list becomes {3, 2, 5, 9, 6, 8}. Implement the method in a way that takes at most **list.length** comparisons. Write a test program that prompts the user to enter a list and displays the list after the partition. Here is a sample run. Note that the first number in the input indicates the number of the elements in the list. This number is not part of the list.

```
Enter list: 8 10 1 5 16 61 9 11 1
After the partition, the list is 9 1 5 1 10 61 11 16
```

Task – 9
(*Sort characters in a string*) Write a method that returns a sorted string using the following header:

public static String sort(String s)

For example, **sort("acb")** returns **abc**.
Write a test program that prompts the user to enter a string and displays the sorted string.

Multi-Dimensional Arrays:

Task – 10
(*Largest block*) Given a square matrix with the elements 0 or 1, write a program to find a maximum square submatrix whose elements are all 1s. Your program should prompt the user to enter the number of rows in the matrix. The program then displays the location of the first element in the maximum square submatrix and the number of the rows in the submatrix. Here is a sample run:

```
Enter the number of rows in the matrix: 5
Enter the matrix row by row:
1 0 1 0 1
1 1 1 0 1
1 0 1 1 1
1 0 1 1 1
1 0 1 1 1
The maximum square submatrix is at (2, 2) with size 3
```

Your program should implement and use the following method to find the maximum square submatrix:

public static int[] findLargestBlock(**int**[][] m)

The return value is an array that consists of three values. The first two values are the row and column indices for the first element in the submatrix, and the third value is the number of the rows in the submatrix.