Assignment One: Using Data Structures

Learning Objectives

- Evaluate your understanding of the different data structures your learned about which include
  - Lists
  - Stacks
  - Queues

Directions

Write a program to solve each of the following programming questions. Submit your code on d2l with the following information

Submission file should be named as follows:
SODV2401_term_assignment1_bvcusername
All files should identify the author with the following information:
Course code : SODV2401
TERM/Year : TermYear
Assignment code: A1
Author :
BVC username :
Date created : YYYY-MM-DD
Description :

Question 1
A student is a person, and so is an employee. Create a class Person that has the data attributes common to both students and employees (name, social security number, age, gender, address, and telephone number) and appropriate method definitions. A student has a grade-point average (GPA), major, and year of graduation. An employee has a department, job title, and year of hire. In addition, there are hourly employees (hourly rate, hours worked, and union dues) and salaried employees (annual salary). Define a class hierarchy and write an application class that you can use to first store the data for an array of people and then display that information in a meaningful way.

Question 2

Develop a program to maintain a list of homework assignments. When an assignment is assigned, add it to the list, and when it is completed, remove it. You should keep track of the due date. Your program should provide the following services:

- Add a new assignment.
- Remove an assignment.
- Provide a list of the assignments in the order they were assigned.

- Find the assignment(s) with the earliest due date.

Question 3

Develop an Expression Manager that can do the following operations:

Balanced Symbols Check

- Read a mathematical expression from the user.
- Check and report whether the expression is balanced.
- {, }, (,), [,] are the only symbols considered for the check. All other characters can be ignored.

Infix to Postfix Conversion

- Read an infix expression from the user.
- Perform the Balanced Parentheses Check on the expression read.
- If the expression fails the Balanced Parentheses Check, report a message to the user that the expression is invalid.
- If the expression passes the Balanced Parentheses Check, convert the infix expression into a postfix expression and display it to the user.
- Operators to be considered are +, –, *, /, %.

Postfix to Infix Conversion

- Read a postfix expression from the user.
- Convert the postfix expression into an infix expression and display it to the user.
- Display an appropriate message if the postfix expression is not valid.
- Operators to be considered are +, –, *, /, %.

Evaluating a Postfix Expression

- Read the postfix expression from the user.
- Evaluate the postfix expression and display the result.
- Display an appropriate message if the postfix expression is not valid.
- Operators to be considered are +, –, *, /, %.
- Operands should be only integers.

Implementation

- Design a menu that has buttons or requests user input to select from all the aforementioned operations.

Question 4

Write a menu-driven program that uses an array of queues to keep track of a group of executives as they are transferred from one department to another, get paid, or become unemployed. Executives within a department are paid based on their seniority, with the person who has been in the department the longest receiving the most money. Each person in the department receives $1000 in salary for each person in her department having less seniority than she has. Persons who are unemployed receive no compensation.

Your program should be able to process the following set of commands:

Join <*person*> <*department*>	<*person*> is added to <*department*>
Quit <*person*>	<*person*> is removed from his or her department
Change <*person*> <*department*>	<*person*> is moved from old department to <*department*>
Payroll	Each executive's salary is computed and displayed by department in decreasing order of seniority

Hint: You might want to include a table that contains each executive's name and information and the location of the queue that contains his or her name, to make searching more efficient.

Rubric/Marking Criteria

Criteria	**Weight/Rating**
Question 1	25%
Question 2	25%
Question 3	25%
Question 4	25%
Total	**100%**