# COMP 2230 - Data Structure, Algorithm Analysis, and Program Design

**Laboratory No. 6**

**Total: 50**

Due date and time for submission: Midnight of Sunday, December 1, 2019

Method of submission: Moodle.tru.ca

This is our last lab. In this lab, we will perform some experiments with Hash Table, Disjoint Set, and Graph algorithms. You may use the source codes that we have created in the lab exercises folder of the Moodle portal and alter the code for this lab.

Please see me or the TA if you are having any difficulty in implementing any of the methods in this lab.

**Q1. [5 marks]** We have discussed Disjoint Set ADT in the lab. It performs "union" and "find" operations. In a simple implementation of union operation, it changes the parent of one set to another set without considering any weight factor of the parent. This can create a skewed tree and search may become inefficient. Create a DisjointSetADT class with "Union" and "Find" operations. In the union operation consider the weight of the parents and make the heavier set as the parent set of the lighter set.

**Q2. [5 Marks]** Given an array of characters, give an O(n) algorithm (that is you can visit each character only once) for removing the duplicates. *[Hint: use hash table].*

**Q3. [10 Marks]** Given a list of pairs; Give an efficient method (O(n)) to print all "Symmetric Pairs". A pair is symmetric if both pair (i, j) and pair (j, i) exist in the list. For example, in { {3, 1}, {2, 6}, {3, 5}, {7, 4}, {5, 3}, {8, 7} } , note that {3, 5} is already present in the list when you encounter {5, 3}, that means they are symmetric pair, so print this pair when you encounter {5, 3}. *[Hint: use hash table]*

Algorithm:

1) Read the pairs of elements one by one and insert them into a hash table. For each pair, consider the first element as a key and second element as value.
2) While inserting the key elements check if the hashing of the second element of the current pair is the same as the first number of the current pair.
3) If they are the same, this means that the pairs are symmetric and output that pair.
4) Otherwise, insert that element into that. That means, use the first number of the pair as key and the second number as value and insert them into the hash table.
5) By the time we complete the scanning of all pairs, we have output all the symmetric pairs.

**Q4. [5 Marks]** Given an unweighted directed graph G, write a program that counts and prints all simple paths from a given ‘s’ to a given ‘d’. Assume the graph G is represented using adjacent matrix.

**Q5 [5 Marks]** Write a program to determine whether a given unweighted undirected graph G contains a cycle or not.

**Q6**. **[5 Marks]** Implement the GraphADT using Adjacency List and test it with BFS search.

**Q7. [15 Marks]** There is a maze of size N x N. You are designing a robot that can traverse through the maze and find the exit point. You can select a starting point in the maze ‘S’ and there is only one exist point marked as ‘E’. Your robot needs to find the shortest path (with minimum energy cost) from 'S' to 'E' in the maze.

- ➔ The robot can only move **left (L), right (R), up (U) and down (D)** from a cell.
- ➔ An empty cell in a maze is represented as **‘.’**
- ➔ Once the robot moves from one cell, it loses one unit of energy.
- ➔ There are some obstacles in some cells (marked as **‘R’**), the robot cannot pass through them.
- ➔ You cannot cross the boundary of the maze.
- ➔ If there is no path then print -1, otherwise print the shortest path (in terms of move) from ‘S’ to ‘E’.

*Sample input and output format (please strictly follow these formats in your code).*

**Sample 1**

`Input:`

`4` (this is the value of the N, which means the maze is a 4x4 maze)

```
. . . S
. R R .
. . R R
R . . E
```

`Output:` 9 moves.

L L L D D R D R R

**Sample 2**

```
5
. . S . R
. R R . .
. . R R E
R . . .  R
R . . .  R
```

Output: 4 moves.

R D R D

**Sample 3**

5
. . S . R
. R R . .
. R . R R
R . . E .
R . . . R

Output: -1

**Note:**

***Make sure, you have exception handling code in place for each exceptional condition for every question. Also, make sure you have tested all the codes with various boundary conditions.***

***Please submit all the .java files and scree captures of test runs of the code. Snapshots of the test run for various data set is very important, please submit that with your source code.***

***Grading Rubric***

| | |
|---|---|
| 1. *Code compiled without any error* | *20%* |
| 2. *Good Comments* | *20%* |
| 3. *All requirements are met* | *40%* |
| 4. *Screenshots of several test runs are included* | *10%* |
| 5. *Apply OOP principals*<br>• *Modularity*<br>• *Good use of classes*<br>• *Encapsulation* | *10%* |

*Note: All problems will be graded based on the above-mentioned rubric.*