CSCI2100A Data Structures
Homework #3

Due: 15th March

Create a directory **Assig#3** and then for each question create a subdirectory under **Assig#3** (e.g. Q1 for question 1). For each programming question, put all the source code and makefile for that question to the question subdirectory. Each program reads data from the input data file named *DataIn*, also located under the same subdirectory.

Algorithms in pseudocode

(Pseudocode shows the statements in similar logic & structures in programming)

Q1. Using only the algorithms in QueueADT, write an algorithm in pseudocode called catQueue that concatenates two queues together. The second queue should be put at the end of the first queue.

Q2. Write the procedures ***ReadPoly(P), WritePoly(P), AddPolys(P1, P2, P),*** using linked list implementation. ***ReadPoly(P)*** reads a polynomial to a linked list and returns it by pointer P. ***WritePoly(P)*** prints out the polynomial pointed by P. ***AddPolys(P1, P2, P)*** adds two polynomials P1 and P2, and returns a new polynomial pointed by P. Define the data types you'll use in your program.

Sample input data format:

```
N   C1   E1   C2  E2   . . .
M   C1   E1   C2  E2   . . .
```

/* N is the # of data items in the first polynomial P1, and followed by the data items in P1. M is the # of data items in the second polynomial P2, and followed by the data items in P2. Each data item has two values C1 E1 - coefficient exponent, i.e.

$4X^3 - 5X$ ➔ 4 3 -5 1 (4 3 the first data item, -5 1 the second). */

Sample output data format:

```
P1:     C1  E1  C2  E2  . . .
P2:     C1  E1  C2  E2  . . .
P:      C1  E1  C2  E2  . . .
```

Programming Questions

Q3. The Bashemin Parking Garage contains a single lane that holds up to ten cars. There is only a single entrance/exit to the garage at one end of the lane. If a customer arrives to pick up a car that is not nearest the exit, all cars blocking its path are moved out, the customer's car is driven out, and the other cars are restored in the same order that they were in originally. Write a C program that processes a group of input lines. Each input line contains an 'A' for arrival or a 'D' for departure, and a license plate number. Cars are assumed to arrive and depart in the order specified by the input. The program should print a message whenever a car arrives or departs. When a car arrives, the message should specify whether or not there is room for the car in the garage. If there is no room, the car leaves without entering the garage. When a car departs, the message should include the number of times that the car was moved out of the garage to allow other cars to depart (hint: use stack to simulate the process). The main program and data types are given for your use, on the course/assignments website.

Sample input data format:		Sample output data format:
A	HP 1256	Enter, please!
A	SK 8510	Enter, please!
D	HP 1256	Moved 0, bye!
. . .		. . .
A	GG 3016	Sorry, no room!
D	SK 8510	Moved 1, bye!
. . .		. . .

Q4. A *deque* is a data structure consisting of a list of items, on which the following operations are possible:

Push(X,D): Insert item X on the front end of *deque* D.
Pop(D): Remove the front item from *deque* D and return it.
Inject(X,D): Insert item X on the rear end of *deque* D.
Eject(D): Remove the rear item from *deque* D and return it.

Write routines to support the *deque* that take O(1) time per operation. Your main program will print out the items in *deque* after each operation. *Deque* can have at most 30 items.

Sample input data format:

```
Nop
Op    X
Op
.  .  .
```

/* Nop is the num of operations to be performed, and Op is the operation # with or without an operand (e.g. 1 – Push, 2 – Pop, 3 – Inject, 4 – Eject). One example of input sequence is shown below:

Sample output data format:

```
5
1    3                    Push:   3
3    7                    Inject: 3  7
2                         Pop:    7
3    2                    Inject: 7  2
4                         Eject:  7
```

Only integer operand is used. */

Total: 60 marks

10 marks for each pseudocode question;
20 marks for each programming question.
For programming questions Q3, Q4:
15 for correctness,
5 for programming style (i.e. comments, indexing & space).

The submission for **Assig#3** will be posted in the website ~csci2100A later.

--- THE END ---