Instructor: Abdulkadir GORUR

Time Allowed: 120 minutes  
Name and Surname: ..............................................
Date: 25/07/ 2007
Student Number: ..............................................

1. **(20p)** Write a function `isNotCovered` that takes `Stack1` and `Stack2` as parameters (containing the same type of elements) and returns `TRUE` if `Stack2` is not contained in `Stack1` preserving the order. (20)

For example, the following would return `FALSE` if passed as parameters to your method:

- Stack1: `[6, 1, 3, 4, 9, 5] top`
- Stack2: `[1, 3, 4] top`

The following parameters would return `TRUE`,

- Stack1: `[1, 3, 4, 9, 5] top`
- Stack2: `[1, 4, 9] top`

- all valid operations for stack and queue are available
- size of stacks are not known. (if you develop algorithm which will work only on given example the solution will not be accepted)

1. **(20p)** Given a Queue `q`, whose elements are in increasing order, write a function that returns the number of items occurring once in the queue. If the queue is `q[front]12, 23, 23, 45, 67, 67, 67, 67, 82[rear]`, it should return 3.
2. (25p) Write a function called `generateString(s1, s2)` that takes two strings as its arguments. The string `s1` contains pairs of characters. A pair is defined as single digit numeric character followed by alpha/numeric character. The function will fill `s2` string by considering the content of `s1`. Following illustrates how `generateString` function generates `s2` based on `s1`. (Assume `s1` is either empty or always contains pairs of characters) (20)

Ex2:  
s1="4a8e3z4b"
     s2="aaaaaeeeeeziizbb"

Ex3:  
s1="2a3151"
     s2="aal11iiii"

Ex4:  
s1="263k5q"
     s2="66kkkqqqqq"

2. Write a function with the signature `removeRepeatedSequence(Queue *q)` that will modify Queue contents by leaving single character for each repeated sequence of characters. Consider following example (20p)

```
q:       Ddddddaaaatttttta       Sttrruuccttuureesss
       front
rear
```

After call to `removeRepeatedSequence` q becomes

```
q:       Data Structures
       front                   rear
```
3. Write a short program that takes an integer value (call it n) from the user and prints out a square of numbers starting with 1 in the top left corner, 2 in a ring around the 1, 3 in a ring around the 2's, For example, if the user enters 3 as the input, your program should print out: (15p)

```
1 2 3
2 2 3
3 3 3
```

If the user enters 5 as the input, your program should print out:

```
1 2 3 4 5
2 2 3 4 5
3 3 3 4 5
4 4 4 4 5
5 5 5 5 5
```
4. Write a function called `catStack`, that concatenates the contents of first stack on top of second. (void `catStack(Stack *s1, Stack *s2)`) (15p)

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

S1

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

S2

S2
5. Write a function that swaps (exchanges) two nodes in a list. The nodes are identified by key values (int) that are passed as parameters together with the head node. (15p)

```
10 0xaa
0xffa

50 0xbbcc
0xaaa

8 0xbacf
0xbbcc

60 NULL
0xbacf
```

If we call the function with following arguments the linked list should become as follows:

```
head=Swap(head,10,8);
```

```
8 0xaaa
0xbbcc

50 0xffa
0xaaa

10 0xbacf
0xffa

60 NULL
0xbacf
```

5. (15p) Given a linked list l, whose elements (integer) are in increasing order, write a function that removes the non repeating numbers from list

```
1->12-> 23-> 23-> 45-> 67-> 67-> 67-> 67-> 82->NULL
```