TIC1002: Introduction to Computing and Programming II
Semester II, 2020/2021

Practical Exam 1 (20th February, 2021)
Time allowed: 1 hour 30 minutes

Task 1. Hmm... That's Kurious? [10 marks]

Dr.Curio found a very curious phenomena. If we start with a positive number **N** with at most 4 **unique digits** (i.e. no duplicate digits), we can perform the following "k-operation":

- Arrange the digits in N in ascending order to get **S** (i.e. **S**mall).
- Arrange the digits in N in descending order to get **B** (i.e. **B**ig).
- We then set **N = B – S** (i.e. N is now a new number)

Strangely enough, if we keep performing the "k-operation" over and over, N will **always become the number 6174** after no more than 7 rounds of "k-operation"!

Below are 3 examples

N= 8352	N= 9189	N = 36 (i.e. 0036)
1st Round:	1st Round:	1st Round:
S = 2358	S = 1899	S = 0036
B = 8532	B = 9981	B = 6300
N = 8532-2358 = 6174	N = 9981 - 1899 = **8082**	N = 6300 - 36 = **6264**
(done after 1 round)	2nd Round:	2nd Round:
	S = 288	S = 2466
	B = 8820	B = 6642
	N = 8820 - 288 = **8532**	N = 6642 - 2466 = **4176**
	3rd Round:	3rd Round:
	S = 2358	S = 1467
	B = 8532	B = 7641
	N = 8532 - 2358 = 6174	N = 7641 - 1467 = 6174
	(done after 3 rounds)	**(done after 3 rounds)**

**As shown in some of the examples, when N is less than 4 digits, you can just assume there are leading '0's in front, e.g. N = 288 ➔ 0288.

To conserve space, we only show examples that are done in 3 rounds or less, you can try N = 2357, which takes the maximum 7 rounds to reach 6174. Also, in case you are wondering, once you reach 6174, "k-operation" will no longer change the N (i.e. you get back 6174!). It is a weird but true phenomena. Surprising, eh?

Your task is described on the next page.

Existing Functions to Help:

There are several functions given in the template code to help your attempt. **You should make use of these functions instead of writing your own.** Note that the parameters are not shown.

`selectionSort()`	The standard selection sort (ascending order) as covered in the lecture.
`numberToArray()`	Convert an integer into an array of individual digits.
`arrayToNumber()`	Convert an array of digits into a single integer.

Implement the following functions:

`void reverseSelectionSort(int a[], int n);          //3 marks`
Sort the array **`a[]`** of size **n** in descending order.
Restriction: You need to follow the selection sort approach taught in the course, but sort the items in descending order.

`int kNumber( int N );                                //7 marks`
N is a positive number with up to 4 unique digits. You can assume N follows the specification.
This function returns the number of rounds of "k-operation" performed on N to reach 6174. **If N is equal to 6174 at the start, return 0.**

For this task, you are **not allowed to write any additional functions**. Only submit the above two functions.

Additional Requirements for kNumber() only (i.e. not applicable to reserveSelectionSort())

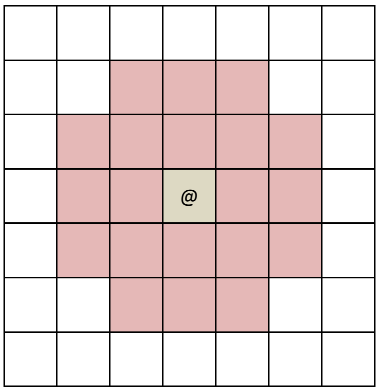
Implementation	Maximum Mark
Iterative (used **any form** of loop)	**5 marks**
Recursive (**NO** loop of **any form**)	**7 marks**

Task 2. Keep your distance! [8 marks]

Ministry of Health (MOH) asked for your help to check for social distancing violation. They plotted the positions of people on a **2D map[] array** using trace together data. Each location is a Boolean value (`true` = occupied by someone, `false` = empty). Below is a sample 10 rows x 10 columns **map[]** data. For ease of visualization, we show occupied locations using the character '@' and show empty locations using '.':

	0	1	2	3	4	5	6	7	8	9
0	@	@	.	.	.	@	.	.	@	@
1	@	.	@	.	.	@	.	.	.	@
2	.	@	@	.	.	.	@	@	.	@
3	.	@	.	.	.	.	.	.	.	.
4	.	.	.	@	@	@	@	@	.	.
5	.	.	.	@	@	@	@	@	.	.
6	.	@	.	@	@	@	@	@	.	.
7	.	.	.	@	@	@	@	@	.	.
8	.	.	.	@	@	@	@	@	.	.
9	.	@	.	.	.	.	.	.	.	.

Given an occupied location '@', MOH considered the shaded locations shown on the right as "too close" (i.e. violate social distancing). It is essentially a "fat cross" shape ➔



You are asked to **count the number of occupied locations in the** "fat cross" shape given a specified location. i.e. essentially checking how many persons violated social distancing **discounting the '@' location itself.** Note that if the specified location is empty, we do not need to check for social distancing violation.

Given the sample map at the start of this question:

Check location (*row*, *col*)	Number of social distancing violations
4, 0	**0**, because location (4, 0) is not occupied!
6, 5	**20**, every locations in the "fat-cross" are occupied!
0, 0	**4**
0, 8	**4**
6, 1	**3**

Implement the following function:

`int violateSD(bool map[][MAXCOL], int row, int col)`
`map[]` is the 2D map. `MAXROW` is defined as 10 and `MAXCOL` is defined as 10. This function return the number of social distancing violation at location (row, col). If the location (row, col) is not occupied, return 0. Note that the location (row, col) itself are not included in the count. (see examples given in previous page if you are not sure).

You should design **appropriate helper function(s)** to help with implementation. **Submit the function above and <u>any new helper function(s).</u>**

2 out of the total 8 marks for this question is awarded for **modularization.**

<u>Good Programming Style [2 marks]</u>

Programming Style	Applicable to	Total
Consistent Indentation	Tasks 1 and 2	1 mark
Good variable naming	Tasks 1 and 2	1 mark

**~~~ End of PE Question ~~~**