TOBB ETU – 2025 Fall
BIL 214 – Exercises 5

Instructions

- Write function definitions (including main) and complete C program code for each question.
- Do not include any additional libraries.
- Do not define constant/global variables. Do not use recursions.
- From Standard C Library Functions, you can only use the ones listed in the worksheet.
- You can assume that parameters are always passed correctly by the user.
- You can assume that the lengths of the arrays are sufficient.
- You can assume that strings end with the null character as expected.
- Do not use integer value of a char directly. For example, do not use 65 for ‘A’.

Questions

soru1. Write function definitions and complete main function of below C file. In main function, call other functions when it is possible. Do not make function calls in other functions. You must follow declaration and loop restrictions to get points from a part.

- **swap** exchanges the values of two integers pointed by x and y. You are not allowed to make any variable declarations except 1 variable.
- **isSorted** checks if integer array A is sorted (in ascending order) or not. If it is, set the integer pointed by “p” to 1. Otherwise set it to 0. You can assume that there is always 0 (stop character) at the end of the array. You are not allowed to make any variable declarations.
- **indexOfMin** finds the minimum value of the positive integer array “A” and saves its index (starting from 0) to integer pointed by “p”. You can assume that there is always 0 (stop character) at the end of the array. You are not allowed to make any variable declarations except 1 pointer.
- **main** gets positive integers from the user one by one and saves them to array “A” until the user enters 0. Add 0 to the end of the array as stopping character. Check if “A” is sorted (in ascending order) or not. If it is not, use bubble sort algorithm to sort “A”. Space complexity of sorting must be O(1) meaning that you cannot use more than constant memory like additional arrays. You are allowed to use at most 3 loops.

Bubble Sort Algorithm:

```
size ← size of A
FOR i = 0 to (size-2)
    Find the index of min element of subarray of A starting from i-th element to the end of the array.
    Swap the minimum element of the subarray (of A) and i-th element of A.
Print sorted array A as a line.
```

Example

```
7
5
8
9
2
8
2
4
6
1
3
2
5
7
0
1 2 2 2 3 4 5 5 6 7 7 8 8 9
```

```
#include <stdio.h>
void swap(int *x, int *y);
void isSorted(int *A, int *p);
void indexOfMin(int *A, int *p);
int main(int argc, char *argv[]) {
        int A[1000];
```

soru2. The function definitions and complete main function of below C file. In main function, call other functions when it is possible. Do not make function calls in other functions. You must follow declaration and loop restrictions to get points from a part.

- **findLength** saves the length of the string "s" to the integer pointed by "p". You are not allowed to make any variable declarations.
- **isAllLowerChars** checks if the string "s" consists of only lower alphabet characters or not. If it is, set the integer pointed by "p" to 1. Otherwise set it to 0. You are not allowed to make any variable declarations.
- **findCharFirstLoc** makes a search on the string "s" and saves the address of the first occurrence of the character "c" to the memory pointed by "p". if there is no such character in the string, assign NULL. You are not allowed to make any variable declarations.
- **main** gets a string and a char from the user and saves them "s" and "c" respectively. Check if s consists of only lower characters and c is also a lower character. If they are not, ask the user to enter s and c again until desired inputs are obtained. You can assume that there is no space character in the string. Split the string into tokens by using char "c" as delimiter and display each token and its length as in the example below. Space complexity of splitting must be O(1) meaning that you cannot use more than constant memory like additional arrays. You are allowed to use at most 2 loops.

Algorithm:

```
Save the address of the beginning of the string into a pointer, call it "p".
REPEAT
        Find the address of the first occurrence of "c".
        Set the value in the found address to the null character.
        Print string starting from the location "p" and its size.
        Set "p" to the address of the character just after the found character "c".
UNTIL no address of "c" is found
Print string starting from the location "p" and its size.
```

Example

```
quizeprogramealleaeanenoteepiclike e
4 quiz
7 program
3 all
1 a
2 an
3 not
0
6 piclik
0
```

```
#include <stdio.h>
void findLength(char *s, int *p);
void isAllLowerChars(char *s, int *p);
void findCharFirstLoc(char *s, char c, char **p);
int main(int argc, char *argv[]) {
        char s[800], c;
```