תרגיל מס' 5 – C מערכים (מתרגל אחראי: דור ניסים)

הוראות הגשה

שאלות בנוגע לתרגיל נא להפנות דרך פורום הקורס שנפתח במיוחד לשם כך:

https://piazza.com/biu.ac.il/fall2018/89110

אם לא נענתה תשובה תוך 24 שעות, נא לשלוח אלי (דור) מייל עם לינק לדיון הרלוונטי ואענה. המייל הוא:nisimdor@gmail.com. בכל מייל יש לציין **שם, שם משתמש, מס' קורס, וקבוצת תרגול.**

- מועד פרסום: 02/12/18
- מועד אחרון להגשה: 16/12/18
- יש לשלוח את הקבצים באמצעות האתר:
  http://help.cs.biu.ac.il/submit.htm
- שם ההגשה של התרגיל: ex5
- יש להקפיד מאוד על כל הוראות עיצוב הקלט והפלט, כמפורט בכל סעיף וסעיף. על הפלט להיראות בדיוק כמו בדוגמאות. **אין להוסיף או להשמיט רווחים או תווים אחרים ואין להחליף אותיות גדולות בקטנות או להיפך ☹ אי-הקפדה על פרטים אלה עלולה לגרור ירידה משמעותית ביותר בציון התרגיל עד כדי 0. ראו הוזהרתם!**
- להזכירכם, העבודה היא אישית. "עבודה משותפת" דינה כהעתקה.
- אין להדפיס למסך שום דבר מעבר למה שנתבקש בתרגיל.
- יש לוודא שהתרגיל מתקמפל ורץ על השרתים באוניברסיטה (u2) ללא שגיאות/אזהרות.
- אתם יכולים לעבוד עם כל עורך טקסטואלי שאתם מעדיפים. להזכירכם – pico בשרתי linux שבמעבדות; notepad ב-windows ; או בסביבת פיתוח ייעודית.

הקפידו על כתיבה לפי קובץ ה-Coding-Style שבאתר הקורס!!

הוראות כלליות לתרגיל

כתבו תכנית בעלת הזרימה הבאה:

```
int mission;
do{
    printf("Please enter the mission number");
    scanf("%d", &mission);
    switch(mission){
        case 1: Mission1();
                break;
        case 2: Mission2();
                break;
        case 3: Mission3();
                break;
        case 4: Mission4();
                break;
        case 5: Mission5();
                break;
    }
}while(mission >=1 && mission <=5);
```

ניתן להניח שכל קלט המתקבל במשימות הינו **בעל הפורמט המתאים** (לדוגמא, אם נאמר כי אתם מקבלים מספר שלם – int – אכן תקבלו מספר שלם).

אולם חובה לבדוק את תקינות הקלט. **במקרה וקלט לא תקין** יש לבצע את ההוראות המפורטות בכל משימה ומשימה. (חזרה לתוכנית הראשית – אלא אם כן נאמר אחרת)

יש להשתמש בלבד בחומר הנלמד עד כה בתרגולים (עד מערכים + רקורסיות כולל).

אסור להשתמש במשתנים גלובליים וסטטיים!!!

- כאשר מבקשים שפונקציה תהיה רקורסיבית, המשמעות היא **אסור לולאות**
- מאחר ולמדנו על פונקציות, נסו לחשוב על פתרון למטלה תוך שימוש בפונקציות.

המטלה כוללת 11 קבצים:

- ex5.c – קובץ ה- main אשר מהווה את התוכנית הראשית של התרגיל (**לא** לשנות את הקובץ! – מלבד שינוי ההערה הקשורה לפרטי מגיש התרגיל)
  הקובץ מכיל לולאה ו- SWITCH CASE אשר בהינתן קלט בין 1 ל- 5 מפעיל פונקציה מתאימה.
  כאשר התוכנית מפעילה את הפונקציה **()**Missioni
  (כאשר $1 \leq i \leq 5$) היא מפעילה את הפונקציה המוגדרת בקובץ Missioni.c
- לכל $1 \leq i \leq 5$ קיימים הקבצים הבאים:
  - Missioni.h – קובץ המכיל את חתימות הפונקציות (מכיל לפחות 2 פונקציות כאשר אחת מהן היא **()**Missioni).
  - Missioni.c – קובץ מימושים עבור הפונקציות. לרוב הפונקציה **()**Missioni תהיה מוממשת, **אולם** אתם יכולים לשנות אותה **<u>לפי ראות עיניכם</u>**.

נא לא לשנות את שמות הקבצים!

- שימו לב כי **קיים קובץ oracle.out** אשר מהווה תוכנית **<u>אשר ניתן להריץ במערכת ה- U2</u>** על מנת שתוכלו להריץ בדיקות על מקרי קצה!

בכל משימה תתבקשו לממש פונקציה (אחת לפחות). המימוש צריך להיות בתוך הפונקציה שמוכנה לכם בקובץ Missioni.c.

לאחר שתממשו, תוכלו לקמפל ולהריץ את התכנית עם ה- main המוכן.
בסיום המטלה הנכם נדרשים להגיש **את כל הקבצים (כל הקבצים שתוארו לעיל)** ובהם המימושים של המשימות.

ניתן להגדיר פונקציות עזר, כאשר בקובץ Missioni.h יופיעו **החתימות של פונקציות העזר** ובקובץ Missioni.c יופיעו המימושים של פונקציות עזר.
<u>לא לשכוח</u> לרשום בהערה מה **הפונקציה אמורה לבצע**. (לפי ה- coding style).

כמו כן, הנכם מתבקשים לרשום **<u>הערה בכל קובץ הכוללת:</u>**

- **שם פרטי ושם משפחה**
- **ת.ז.**
- **שם משתמש בסאמביט**
- **ex5**

Semi-Similar Strings – משימה 1

In this mission you need to implement the function in **Mission1.c**:

```
int isSemiSimilar(char target[], char source[])
```

Two same length strings (**a** and **b)** are called **Semi-similar strings** if string **a** is equal to string **b** after switching characters in **b**.

Consider the following example where string a is **abcdefg** and string b is **cdfegba**. It is clear that if we switch the characters in string **b** we will get string **a**.

You will be given **n** pairs of strings, and you must output the amount of pairs that are Sem-similar.

Input Format

The input format will be as follows:

```
Please enter the amount of pairs:
<n>
Please enter the first string of pair 1:
<Str1_1>
Please enter the second string pair 1:
<Str2_1>
…..
Please enter the first string of pair <n>:
<Str1_n>
Please enter the second string pair <n>:
<Str2_n>
```

Please make sure that the output messages are as described above. (The system is case-sensitive).

- <**Str1_i**> - The first string of pair **i** to receive as input
- < **Str2_i** > - The second string of pair **i** to receive as input

Note: Two string with different length aren't considered **Semi-similar strings**.

Constraints

- $1 \leq |String| \leq 10^4$ (The length of the string includes \0).
    - You can assume that the string length will **not** exceed the limit

  Each string will contain only letters from ***'a'-'z'***.
    - If the string includes a char not in the given range, **you must end the mission** and return to the main menu (Main program- "Mission selection").

- 1 <= *n* <= 100
    - If **n** is not in the given range, **<u>you must end the mission</u>** and return to the main menu (Main program- "Mission selection").

Output Format

The output format will be as follows:

```
The amount of Semi-Similar strings is < result><newline>
```

- **< result>** - Integer that indicates how many pairs are **Semi-similar.**

Sample Input

```
Please enter the amount of pairs:
3
Please enter the first string of pair 1:
abcdefg
Please enter the second string of pair 1:
cdfegba
Please enter the first string of pair 2:
gfeg
Please enter the second string of pair 2:
geff
Please enter the first string of pair 3:
abcd
Please enter the second string of pair 3:
abcde
```

Sample Output

```
The amount of Semi-Similar strings is 1<newline>
```

Explanation

The first pair corresponds to the example given in the Problem Definition, and the pair is indeed **Semi-similar**.
In the second pair, the strings are the same length but there is only one **'f'** character in the target string and two **'f'** characters, and thus the pair is not **Semi-similar**.

In the last pair, the strings are not equal length, and thus the pair is not **Semi-similar**.

In total, there is only **one** pair that has **Semi-similar** strings.

The Cookie Monster Cometh – 2 משימה

In this mission you need to implement the function in **Mission2.c**:

```
int SweetCookies(int cookies[], int n, int K)
```

Cookie monster loves cookies. He wants the sweetness of all his cookies to be greater than value **K**. To do this, Cookie monster repeatedly mixes two cookies with the least sweetness. He creates a special combined cookie with:

$$sweetness = (1 \times Least\ sweet\ cookie + 2 \times 2nd\ least\ sweet\ cookie).$$

He repeats this procedure until all the cookies in his collection have a $sweetness \geq \boldsymbol{K}$.

You are given Cookie monster's cookies. Print the number of operations required to give the cookies a $sweetness \geq \boldsymbol{K}$.

Print **-1** if this isn't possible.

Input Format

The input format will be as follows:

```
Please enter the amount of cookies:
<N>
Please enter the minimum required sweetness:
< sweet >
Please enter cookie number 1:
<A1>
….
Please enter cookie number n:
<An>
```

- <**N**> - The number of cookies
- <**sweet**> - The minimum required sweetness
- <**A**$_i$> - The sweetness of the i^{th} cookie in Cookie monster's collection. **(The Array is not always sorted!)**

Constraints

$1 \leq N \leq 1000$

$0 \leq K \leq 10^4$

$0 \leq A_i \leq 10^5$

If the input **is not in the given range**, you must end the mission and return to the main menu (Main program- "Mission selection").

Output Format

The output format will be as follows:

```
The number of operations that are needed is <result><newline>
```

- **<result> -** the number of operations that are needed to increase the cookie's $sweetness \geq \boldsymbol{K}$.
  Output **-1** if this isn't possible.

Sample Input

```
Please enter the amount of cookies:
6
Please enter the minimum required sweetness:
7
Please enter cookie number 1:
2
Please enter cookie number 2:
1
Please enter cookie number 3:
3
Please enter cookie number 4:
9
Please enter cookie number 5:
10
Please enter cookie number 6:
12
```

Sample Output

```
The number of operations that are needed is 2<newline>
```

Explanation

Combine the first two cookies to create a cookie with $sweetness = 1 \times 1 + 2 \times 2 = 5$
After this operation, the cookies are $3, 5, 9, 10, 12$.
Then, combine the cookies with sweetness **3** and sweetness **5**, to create a cookie with resulting $sweetness = 1 \times 3 + 2 \times 5 = 13$.
Now, the cookies are $9, 10, 12, 13$.
All the cookies have a $sweetness \geq \mathbf{7}$..

Thus, **2** operations are required to increase the sweetness.

Game of Thrones – משימה 3

In this mission you need to implement the function in **Mission3.c**:

```
void CanAnagram(char str[])
```

Dothraki are planning an attack to usurp King Robert's throne. King Robert learns of this conspiracy from Raven and plans to lock the single door through which the enemy can enter his kingdom.

But to lock the door he needs a key that is an anagram of a certain palindrome string.

The king has a string composed of lowercase English letters. Help him figure out whether any anagram of the string can be a palindrome or not.

Input Format

The input format will be as follows:

```
Please enter the string:
<str>
```

<str> - Contains the input string.

Constraints

$1 \leq length\ of\ string \leq\ 100$ (The length of the string includes \0).

- You can assume that the string length will **not** exceed the limit

Each character of the string is a lowercase English letter.

- **If the strings doesn't contain letters from a to z,** you must end the mission and return to the main menu (Main program- "Mission selection").

Output Format

The output format will be as follows:

- If the string can be used as a palindrome

```
The string <str> has the palindrome <pol><newline>
```

<**pol**> - Is a palindrome that is an anagram of the string **<str**>.
The palindrome must have the **smallest lexicographical order** as possible

- If the string does not have any anagram that can be a palindrome

```
The string <str> does not have a palindrome<newline>
```

Sample Input

```
Please enter the string:
aaabbbb
```

Sample Output

```
The string aaabbbb has the palindrome abbabba <newline>
```

Explanation

A palindrome permutation of the given string is *abbabba*.

Sample Input

```
Please enter the string:
aaaccbbbb
```

Sample Output

```
The string aaaccbbbb has the palindrome abbcacbba <newline>
```

Explanation

A palindrome permutation of the given string is abbcacbba.

Sample Input

```
Please enter the string:
aaacccbbbb
```

Sample Output

```
The string aaacccbbbb does not have a palindrome <newline>
```

משימה 4- Suicide Squad

In this mission you need to implement the function in **Mission4.c**:

```
int getMinLength(char mat[][?], int r, int c, int goalR, int goalC, int n)
void loadBoard(char mat[][?], int rows, int cols)
```

You are asked to make an implementation of the game of life by John Horton Conway on a finite torus board and provide its output after c iterations.

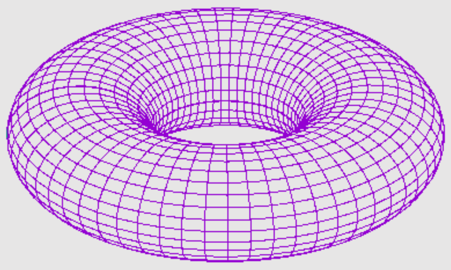
We have a finite torus board of square cells, each of which is in one of two possible states, alive or dead, (or populated and unpopulated, respectively). Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbors dies, as if by underpopulation.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by overpopulation.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Standard input

The input format will be as follows:

```
Please enter the number of rows (n):
<n>
Please enter the number of columns (m):
<m>
Please enter the number of iterations (c):
<c>
Please enter the board:
…
```



- <***n***> - The number of rows in board
- <***m***> - The number of columns in board.
- <***c***> - The number of iterations that you must simulate.

If the input **is not in the given range**, you must end the mission and return to the main menu (Main program- "Mission selection").

On the following *n* lines there will be *m* characters, either * or -, each one representing the value of each cell of the board. * represents a *populated* cell and - an *unpopulated* one.
You can assume that he input board is valid
Please note that the bottom neighbours of the last line are cells in the top line, and the left neighbours of the first column are the cells of the last column.

Standard output

The output format will be as follows:

```
The board after <c> iterations is: <newline>
…..
```

On the output there should be n lines of m characters each, which represent the state of the board after **c** iterations.

You can assume that the board will be valid, i.e. if the board is 3x4 you will receive 3 strings with length of 4 characters each.

Constraints

- $3 \leq n,m \leq 25$
- $1 \leq c \leq 10^5$

If the input **is not in the given range**, you must end the mission and return to the main menu (Main program- "Mission selection").

Pay attention: If a population is extinct (i.e. all the cells are in a state of **dead**) in an iteration, you must **stop** the simulation and display the output message!

For example, if you are given the following board:

```
---
-*-
---
```

And c=100, you must output the following line:

```
The board after 1 iterations is:
---
---
---
```

And NOT the following output:

```
The board after 100 iterations is:
---
---
---
```

Input	Output
Please enter the number of rows (n): 4 Please enter the number of columns (m): 6 Please enter the number of iterations (c): 3 Please enter the board: ------ ------ ------ -***--	The board after 3 iterations is: --*--- ------ --*--- --*---
Please enter the number of rows (n): 5 Please enter the number of columns (m): 6 Please enter the number of iterations (c): 1234 Please enter the board: ------ --*--- ------ -***-- ------	The board after **9** iterations is: ------ ------ ------ ------ ------

```
Please enter the number of rows (n):
7
 Please enter the number of columns (m):
9
 Please enter the number of iterations (c):
5
Please enter the board:

---------

---------

---------

--*****--

---------

---------

---------
```

```
The board after 5
iterations is:

----*----

---***---

--*-*-*--

-***-***-

--*-*-*--

---***---

----*----
```

```
Please enter the number of rows (n):
4
 Please enter the number of columns (m):
6
 Please enter the number of iterations (c):
1
Please enter the board:

-----*

--*---

*-----

-***-*
```

```
The board after 1
iterations is:

**-**-

------

*--*--

-**-**
```

Note: You must use recursion! (NO LOOPS!)

In this mission you need to implement the function in **Mission5.c**:

```
int getMinLength(char mat[][?], int r, int c, int goalR, int goalC, int n)
void loadCastleBoard(char mat[][?], int rows, int cols)
```

You are given a square grid with some cells open (-) and some blocked (X). Your playing piece can move along any row or column until it reaches the edge of the grid or a blocked cell. Given a grid, a start and an end position, determine the number of moves it will take to get to the end position.

You can only move Up, Down, Left & Right (No diagonal movement!)

Input Format
The input format will be as follows**:**

```
Please enter the number of rows and columns (n):
<n>
Please enter the X of the starting position:
<startX>
Please enter the Y of the starting position:
<startY>
Please enter the X of the goal position:
<goalX>
Please enter the Y of the goal position:
<goalY>
Please enter the grid:
….
```

Constraints

- $1 \leq n \leq 25$
- $0 \leq startX,\ \ startY,\ \ goalX,\ \ goalY < n$

    - If the input **is not in the given range**, you must end the mission and return to the main menu (Main program- "Mission selection").

Output Format

The output format will be as follows:

```
The minimum number of steps is <result><newline>
```

- <**result**> **-** An integer denoting the minimum number of steps required to move the castle to the goal position.

If not path is found, print:

```
No path was found from (<startX>, <StartY>) to (<goalX>, <goalY>) <newline>
```

Sample Input

For example, you are given a grid with sides $n=3$ described as follows:

```
Please enter the number of rows and columns (n):
3
Please enter the X of the starting position:
0
Please enter the Y of the starting position:
0
Please enter the X of the goal position:
1
Please enter the Y of the goal position:
2
Please enter the grid:
---
-X-
---
```

Sample Output

```
The minimum number of steps is 3<newline>
```

Explanation

Your starting position $(startX,Y)=(0,0)$ so you start in the top left corner. The ending position is $(goalX,)=(1,2)$. The path is $(0,0)\rightarrow(0,1)\rightarrow(0,2)\rightarrow(1,2)$. It takes 2 moves to get to the goal.

Sample Input

```
Please enter the number of rows and columns (n):
3
Please enter the X of the starting position:
0
Please enter the Y of the starting position:
0
Please enter the X of the goal position:
```

```
0
Please enter the Y of the goal position:
2
Please enter the grid:
-X-
-X-
---
```

Sample Output

```
The minimum number of steps is 6<newline>
```

Explanation

Here is a path that one could follow in order to reach the destination in steps: $(0, 0) \rightarrow (1, 0) \rightarrow (2, 0) \rightarrow (2, 1) \rightarrow (2, 2) \rightarrow (1, 2) \rightarrow (0, 2)$

Sample Input

```
Please enter the number of rows and columns (n):
3
Please enter the X of the starting position:
0
Please enter the Y of the starting position:
0
Please enter the X of the goal position:
0
Please enter the Y of the goal position:
2
Please enter the grid:
-X-
-X-
-X-
```

Sample Output

```
No path was found from (0,0) to (0,2)<newline>
```

Explanation

There is no valid path from (0,0) to (0,2).

```
Please enter the mission number:
1
Please enter the amount of pairs:
3
Please enter the first string of pair 1:
abcdefg
Please enter the second string pair 1:
cdfegba
Please enter the first string of pair 2:
gfeg
Please enter the second string pair 2:
geff
Please enter the first string of pair 3:
abcd
Please enter the second string pair 3:
abcde
The amount of Semi-Similar strings is 1
Please enter the mission number:
2
Please enter the amount of cookies:
6
Please enter the minimum required sweetness:
7
Please enter cookie number 1:
2
Please enter cookie number 2:
1
Please enter cookie number 3:
3
Please enter cookie number 4:
9
Please enter cookie number 5:
10
Please enter cookie number 6:
12
The number of operations that are needed is 2
Please enter the mission number:
3
```

```
Please enter the string:
aaabbbb
The string aaabbbb has the palindrome abbabba
Please enter the mission number:
3
Please enter the string:
aaaccbbbb
The string aaaccbbbb has the palindrome abbcacbba
Please enter the mission number:
3
Please enter the string:
aaacccbbbb
The string aaacccbbbb does not have a palindrome
Please enter the mission number:
4
Please enter the number of rows (n):
4
Please enter the number of columns (m):
6
Please enter the number of iterations (c):
3
Please enter the board:
------
------
------
-***--
The board after 3 iterations is:
--*---
------
--*---
--*---
Please enter the mission number:
4
Please enter the number of rows (n):
5
Please enter the number of columns (m):
6
Please enter the number of iterations (c):
```

```
123
Please enter the board:
------
--*---
------
-***--
------
The board after 9 iterations is:
------
------
------
------
------
Please enter the mission number:
4
Please enter the number of rows (n):
7
Please enter the number of columns (m):
9
Please enter the number of iterations (c):
5
Please enter the board:
---------
---------
---------
--*****--
---------
---------
---------
The board after 5 iterations is:
----*----
---***---
--*-*-*--
-***-***-
--*-*-*--
---***---
----*----
Please enter the mission number:
```

```
4
Please enter the number of rows (n):
4
Please enter the number of columns (m):
6
Please enter the number of iterations (c):
1
Please enter the board:
-----*
--*---
*-----
-***-*
The board after 1 iterations is:
**-**-
------
*--*--
-**-**
Please enter the mission number:
5
Please enter the number of rows and columns (n):
3
Please enter the X of the starting position:
0
Please enter the Y of the starting position:
0
Please enter the X of the goal position:
1
Please enter the Y of the goal position:
2
Please enter the grid:
---
-X-
---
The maximum number of steps is 3
Please enter the mission number:
5
Please enter the number of rows and columns (n):
3
```

```
Please enter the X of the starting position:
0
Please enter the Y of the starting position:
0
Please enter the X of the goal position:
0
Please enter the Y of the goal position:
2
Please enter the grid:
-X-
-X-
---
The maximum number of steps is 6
Please enter the mission number:
5
Please enter the number of rows and columns (n):
3
Please enter the X of the starting position:
0
Please enter the Y of the starting position:
0
Please enter the X of the goal position:
0
Please enter the Y of the goal position:
2
Please enter the grid:
-X-
-X-
-X-
No path was found from (0,0) to (0,2)
Please enter the mission number:
6
```