פרויקט מסכם תכנות מתקדם – תש"פ סמסטר ב

נתון לוח משחק מ-N שורות ו-M עמודות באשר N,M<9. עבור לוח המשחק שנתון בציור N=4 ו-M=5. מיקום משבצת מתואר ע"י מערך בן שני תוים – אות לשורה ומספר לעמודה. בדוגמא שבציור העמודות ממוספרות מ-1 עד 5 והשורות מסומנות A עד D. לדוגמא: המיקום של המשבצת שמכילה X בלוח הבא מסומן ע"י B4 ומיקום המשבצת שמכילה Z הוא A1.

יש להגדיר את M ואת N בעזרת #define (התחילו מ-4 שורות ו-5 עמודות)

	1	2	3	4	5
A	**Z**				
B		2	2	X	
C		2			1
D				1	

על-מנת לייצג משבצת בלוח המשחק, עושים שימוש בהגדרה הבאה:

```
typedef char boardPos[2];
```

על-מנת לייצג **תזוזה** על הלוח עושים שימוש בהגדרה הבאה:

```
typedef struct _move {
        char rows, cols;
} Move;
```

rows מכיל כמה שורות עברנו בתזוזה - מספר חיובי מייצג תזוזה למטה ומספר שלילי מייצג תזוזה מעלה.
cols מכיל כמה עמודות עברנו בתזוזה - מספר חיובי מייצג תזוזה ימינה ומספר שלילי מייצג תזוזה שמאלה.
שימו לב, הטיפוס char משמש כטיפוס נומרי המכיל מספר בטווח 128- עד 127+.

על-מנת לייצג מערך של תזוזות בלוח עושים שימוש בהגדרה הבאה:

```
typedef struct _movesArray {
        unsigned int   size;
        Move           *moves;
} movesArray;
```

על-מנת לייצג מערך של מיקומי משבצות בלוח עושים שימוש בהגדרה הבאה:

```
typedef struct _boardPosArray {
        unsigned int   size;
        boardPos       *positions;
} boardPosArray;
```

סעיף 1

בסעיף זה יש לכתוב את הפונקציה:

boardPosArray ** validMoves(movesArray **moves, char **board)

הפונקציה מקבלת:
moves הינו מערך דו-מימדי בגודל לוח משחק. בכל תא נתון מערך של תזוזות אפשריות מאותו תא.

Alon Schclar

board הינו לוח משחק בגודל זהה לזה של moves. כל תא יכול להכיל רווח או את התו '*' באשר תא שמכיל רווח הוא תא שניתן לזוז אליו ותא שמכיל '*' הוא תא שלא ניתן לזוז אליו.

הפונקציה מסירה מכל תא במערך moves את התזוזות אשר אינן חוקיות. תזוזה אינה חוקית אם היא יוצאת מגבולות הלוח או שהיא מגיעה לתא אשר מכיל '*'.

הפונקציה מחזירה מערך אותו יש להקצות דינמית בגודל זהה לזה של moves ו-board. כל תא במערך המוחזר הוא מערך של המיקומים החוקיים שאליהם ניתן לעבור מתא זה.

סעיף 2

על-מנת לייצג רשימה של תזוזות, עושים שימוש ברשימה מקושרת דו-כיוונית שמוגדרת כדלקמן:

```
typedef struct _moveCell {
   Move                 move;
   struct _moveCell     *next, *prev;
} moveCell;

typedef struct _movesList {
   moveCell    * head;
   moveCell    * tail;
} movesList;
```

כתבו את הפונקציה:

int display(movesList *moves_list, boardPos start, char **board)

הפונקציה מציגה על המסך את רצף התזוזות אשר מתחיל ב-**start** ומתקדם בהתאם לתזוזות אשר שמורות ברשימה **moves_list** באופן הבא:

הפונקציה תצייר לוח משחק על המסך (יש לכם את החופש לקבוע כיצד יראה הלוח).

במיקום שמיוצג ב-**start** ייכתב #. ניתן להניח כי תא ההתחלה אינו מכיל '*'.

במשבצת הראשונה שאליה עוברים ייכתב המספר 1, במשבצת שעוברים אליה בשלב הבא, ייכתב 2, וכך הלאה. ברשימה הנתונה ייתכן מצב שבו התזוזה עוברת לתא שכבר ביקרנו בו או תא שמכיל '*' – במקרה זה, אין לעבור לתא ויש למחוק מהרשימה את התזוזה הזו.

הערך שיוחזר מהפונקציה הוא מספר התאים שנמחקו מהרשימה.

אין לשנות את התוכן של הארגומנט **board**.

להלן דוגמא ללוח מודפס עבור הרשימה (נקודת ההתחלה היא A2):

(0,2)→<u>(0,-3)</u>→(2,-2)→(0,1)→(3,1)→<u>(-2,0)</u>→(-1,-3)→(-4,2)→(4,1)→(1,-3)→<u>(-1,0)</u>→(-4,1)→(1,2)

	1	2	3	4
A	*	#	6	1
B		9		
C		2	3	10
D	*			*
E	5			7
F	8			4

בסוף הריצה תוכן הרשימה המקושרת יהיה (התאים המודגשים ברשימה לעיל נמחקו):

(0,2)→(2,-2)→(0,1)→(3,1)→(-1,-3)→(-4,2)→(4,1)→(1,-3)→(-4,1)→(1,2)

שימו לב: אין להשתמש בפונקציה gotoxy משום שהיא אינה קיימת ב-MAMA.

Alon Schclar

סעיף 3

נתונות ההגדרות הבאות עבור עץ אשר מתאר את כל מסלולי התנועה האפשריים החל ממיקום התחלתי שנתון בשורש ובהתאם לרשימת תזוזות אפשריות שנתונה בדומה לסעיף 1 (עליכם למצוא בעצמכם מה יש לרשום במקום שבו רשומות 3 נקודות על-מנת שההגדרות יעברו קומפילציה. נושא זה קרוי forward declaration):

. . .

```
typedef struct _treeNode{
   boardPos              position;
   treeNodeListCell   *next_possible_positions;  // רשימת מיקומים
} treeNode;

typedef  struct  _treeNodeListCell {
   treeNode                        * node;
   struct _treeNodeListCell   * next;
} treeNodeListCell;

typedef  struct  _pathTree {
   treeNode    * head;
} pathTree;
```

צומת X בעץ מייצג משבצת בלוח המשחק. ילדיו של הצומת X הם כל המיקומים האפשריים אליהם ניתן להמשיך בצעד אחד מ-X על-פי מערך התזוזות האפשריות שנתון עבור X שיתקבל כפרמטר. הילדים של X נתונים ברשימה המקושרת שראשה נתון ב-next_possible_positions .

המסלולים שבעץ מהשורש לצאצאיו מייצגים מסלולי תזוזות בלוח המשחק אשר מתחילים בצומת אשר נתון בשורש.

כדי להימנע ממצב שבו מסלול יכיל מעגלים, אסור במסלול מסוים לחזור אל משבצת שהופיעה כבר במסלול, כלומר, אסור שרשימת הילדים של הצומת X תכיל מיקומים אשר מופיעים כבר במסלול שמתחיל בשורש ומגיע ל-X.

יש לכתוב את הפונקציה:

pathTree findAllPossiblePaths (boardPos start, movesArray **moves, char **board)

הפונקציה מקבלת:

start – מיקום התחלתי

moves – הינו מערך דו-מימדי בגודל לוח משחק. בכל תא נתון מערך של תזוזות אפשריות מאותו תא.

board – הינו לוח משחק בגודל זהה לזה של moves. כל תא יכול להכיל רווח או את התו '*' באשר תא שמכיל רווח הוא תא שניתן לזוז אליו ותא שמכיל '*' הוא תא שלא ניתן לזוז אליו.

הפונקציה מסירה מכל תא במערך moves את התזוזות אשר אינן חוקיות. תזוזה אינה חוקית אם היא יוצאת מגבולות הלוח או שהיא מגיעה לתא אשר מכיל '*'.

אח"כ, הפונקציה בונה עץ שיכיל את כל המסלולים האפשריים אשר מתחילים ב-**start**. בכל צומת X הפונקציה תחפש את כל הצמתים שניתן להמשיך אליהם בצעד <u>אחד</u> בהתאם לתזוזות שנתונות בתא המתאים ל-X ב-**moves** תוך הימנעות מהליכה במעגלים. את כל צמתי ההמשך הללו (צעד אחד) יש לשמור ברשימה ולאחסנם בצומת X. יש להמשיך ולמצוא את המשכי המסלולים מכל אחד מילדים, ילדי הילדים וכן הלאה. אם לא ניתן להמשיך לאף צומת מ- X אזי X נמצא בסוף אחד מהמסלולים אשר החלו בשורש והוא יהיה עלה בעץ המוחזר.

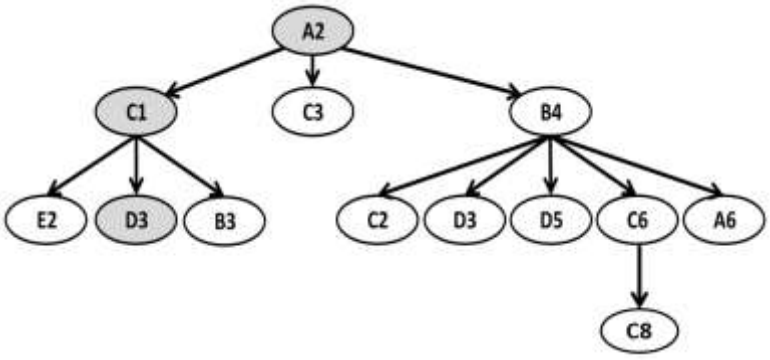
<u>לדוגמא</u>**:**

נתון לוח בגודל 5 שורות ו-8 עמודות.

בפרמטר board המשבצות A4, B8, D6, E1 מכילות *.

Alon Schclar

הפרמטר moves מכיל את המהלכים הבאים:
ממשבצת A2 ניתן לעבור למשבצות C1,C3,B4,E1
ממשבצת C3 ניתן לעבור למשבצות A2,B8,D6
ממשבצת B4 ניתן לעבור למשבצות C2,D3,D5,C6,A6,E1
ממשבצת C1 ניתן לעבור למשבצות E2,D3,B3,C1,D6
ממשבצת C6 ניתן לעבור למשבצות B4,A4,C8
יתר המשבצות לא מכילות תזוזות.
נניח כי תוכן start הוא A2. העץ הבא מתאר את כל המסלולים באורך 3:



המסלול שבאפור מסומן בחיצים בלוח הבא (ההדפסה היא להמחשה בלבד ואינה נחוצה בסעיף זה):

	1	2	3	4	5	6	7	8
A				*				
B								*
C								
D						*		
E	*							

<u>סעיף 4</u>

בסעיף זה עליכם לכתוב את הפונקציה:

movesList *findPathCoveringAllBoard(boardPos start, movesArray **moves, char **board)

הפונקציה מקבלת:
start – מיקום התחלתי
moves הינו מערך דו-מימדי בגודל לוח משחק. בכל תא נתון מערך של תזוזות אפשריות מאותו תא.
board הינו לוח משחק בגודל זהה לזה של moves. כל תא יכול להכיל רווח או את התו '*' באשר תא שמכיל רווח הוא תא שניתן לזוז אליו ותא שמכיל '*' הוא תא שלא ניתן לזוז אליו.
הפונקציה מסירה מכל תא במערך moves את התזוזות אשר אינן חוקיות. תזוזה אינה חוקית אם היא יוצאת מגבולות הלוח או שהיא מגיעה לתא אשר מכיל '*'.

אח"כ, על הפונקציה למצוא מסלול שמכסה את **<u>כל</u>** משבצות הלוח למעט משבצות שמכילות '*', החל מ-**start** ובהתאם למהלכים האפשריים שנתונים ב-**moves**.
המסלול המוחזר ייוצג כרשימה מקושרת של תזוזות החל מ-**start**. התא הראשון ברשימה יכיל את התזוזה מ-**start** אל המשבצת השניה במסלול. יתר התאים ברשימה יכילו את התזוזות אל התאים הבאים במסלול לפי סדרם. במידה ואין מסלול אשר עובר דרך כל משבצות הלוח, הפונקציה תחזיר NULL.

Alon Avitar

סעיף 5

הוחלט לשמור באופן חסכוני סדרת מיקומי לוח בקובץ בינארי באופן הבא: כל מיקום ייוצג ע"י 6 ביטים. 3 הביטים הראשונים ייצגו את מספר השורה פחות 1 ו-3 הביטים הבאים ייצגו את מספר העמודה פחות 1. המיקומים יישמרו ברצף כדי לחסוך במקום. בתחילת הקובץ, יישמר short שיכיל את מספר המיקומים בסדרה. להלן דוגמא לאופן שמירת מיקומים (הרווחים הם למטרת נוחות התצוגה בלבד ואינם חלק מהקובץ):

$$\underbrace{00000000\ 00000101}_{5\ positions}\ \underbrace{010}_{C}\ \underbrace{100}_{5}\ \underbrace{000}_{A}\ \underbrace{011}_{4}\ \underbrace{001}_{B}\ \underbrace{010}_{3}\ \underbrace{011}_{D}\ \underbrace{000}_{1}\ \underbrace{100}_{E}\ \underbrace{010}_{3}\ 00$$

המיקום הראשון ישמר ב-most significant bits ויתר המיקומים ישמרו אחריו. ביטים שנותרים לא מנוצלים יכילו אפסים ב-least significant bits של הבית האחרון. בדוגמא לעיל הקובץ יכיל 6 בתים.

יש לכתוב את הפונקציה:

void saveListToBinFile(char *file_name, boardPosArray *pos_arr)

אשר מקבלת שם של קובץ בינרי ומערך מיקומים ושומרת אותם באופן המתואר לעיל.

סעיף 6

בסעיף זה יש לכתוב את הפונקציה:

int checkAndDisplayPathFromFile (char *file_name, movesArray **moves, char **board)

הפונקציה מקבלת:

file_name - שם של קובץ בינארי אשר מכיל רשימה של מיקומים בפורמט שצוין בסעיף 5.

moves הינו מערך דו-מימדי בגודל לוח משחק. בכל תא נתון מערך של תזוזות אפשריות מאותו תא.

board הינו לוח משחק בגודל זהה לזה של moves. כל תא יכול להכיל רווח או את התו '*' באשר תא שמכיל רווח הוא תא שניתן לזוז אליו ותא שמכיל '*' הוא תא שלא ניתן לזוז אליו.

הפונקציה קוראת את הרשימה ומבצעת את הפעולות הבאות:

אם סדרת המיקומים שנתונה בקובץ אינה מכילה מסלול חוקי על-פי התזוזות שמפורטות ב-moves ועל-פי מה שצוין ב-board, אזי הפונקציה מסיימת ומחזירה את הערך 1.

אחרת, הפונקציה מדפיסה לוח משחק עם המסלול בעזרת הפונקציה מסעיף 2. במידה והמסלול מכיל מיקומים שכבר ביקרנו בהם, יש להתעלם מהם.

אם המסלול החוקי מכסה את כל משבצות הלוח למעט משבצות שמכילות '*', הפונקציה מחזירה 2, אחרת הפונקציה מחזירה 3.

אם הקובץ לא קיים, הפונקציה מחזירה 1-.

הנחיות כלליות:

יש להקפיד על יעילות וחסכון בזמן ריצה וזיכרון.

יש להקפיד גודל פונקציה לא יחרוג ממסך אחד.

יש לפנות זיכרון שהוקצה דינמית ואשר אין בו צורך יותר.

יש להשתמש בdefine# היכן שנחוץ.

Alon Schclar