

פרויקט מסכם תכנות מתקדם

תאריך הגשה: 25.6

בלוח שחמט יש שמונה שורות ושמונה עמודות של משבצות. מיקום משבצת מתואר ע"י מערך בגודל שני תווים – אות לשורה (A עד H) ומספר לעמודה (1 עד 8). לדוגמא: המיקום של המשבצת שמכילה X בלוח הבא מסומן ע"י B4 ומיקום המשבצת שמכילה Y הוא F1.

	1	2	3	4	5	6	7	8
A	Z							
B			2	X				
C		2			1		1	
D				1				1
E						K		
F	Y			1				1
G					1		1	
H								

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

```
typedef char chessPos[2];
```

שימו-לב כי זה מערך בגודל ידוע והוא בן שני בתים. זו אינה מחרוזת ולכן אין \0 בסופה.

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

```
typedef struct _chessPosArray {  
    unsigned int    size;  
    chessPos  *positions;  
} chessPosArray;
```

פרש (knight) הוא אחד הכלים במשחק השחמט. פרש יכול לנוע ממיקומו הנוכחי אך ורק לאחד משמונת המיקומים הבאים:

- 2 שורות מעלה או מטה ועמודה אחת ימינה או שמאלה
 - 2 עמודות ימינה או שמאלה ושורה אחת מעלה או מטה
- לפרש אסור לצאת מגבולות הלוח.

לדוגמא:

בלוח המופיע לעיל, לפרש אשר ממוקם במשבצת אשר מכילה K מותר לעבור אך ורק לאחת מהמשבצות המכילות 1. לפרש אשר ממוקם במשבצת אשר מכילה Z מותר לנוע רק לאחת מהמשבצות שמכילות 2.

סעיף 1

בסעיף זה יש לכתוב את הפונקציה:

```
chessPosArray *** validKnightMoves ( )
```

הפונקציה מחזירה מערך דו-מימדי דינאמי (מערך של מערכים) של מצביעים למערכי מיקומים אפשריים לתנועת פרש. כל תא בשורה r ובעמודה c יכיל מצביע למערך של מיקומים אליהם מותר לפרש לנוע מתא זה.

סעיף 2

על-מנת לייצג רשימה של משבצות, עושים שימוש ברשימה מקושרת חד-כיוונית שמוגדרת כדלקמן:

```
typedef struct _chessPosCell {
    chessPos    position;
    struct _chessPosCell *next;
} chessPosCell;

typedef struct _chessPosList {
    chessPosCell *head;
    chessPosCell *tail;
} chessPosList;
```

כתבו את הפונקציה:

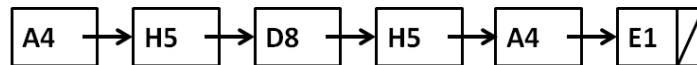
```
void display( chessPosList *lst)
```

המקבלת רשימה של משבצות.

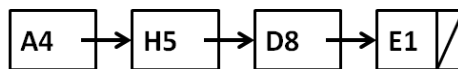
ראשית, הפונקציה מסירה מהרשימה משבצות אשר חוזרות על עצמן. במקרה שמיקום חוזר על עצמו, המופע הראשון שלו מתחילת הרשימה הוא זה שיושאר וכל המופעים אשר נמצאים אחריו ימחקו.

לדוגמא:

אם הקלט לפונקציה יהיה



לאחר הסרת המשבצות אשר חוזרות על עצמן, הרשימה תהפוך ל:



לאחר הסרת המשבצות אשר חוזרות על עצמן, הפונקציה תציג את המשבצות שנותרו ברשימה על המסך באופן הבא:

נניח כי מספר המיקומים ברשימה הוא N . הפונקציה תצייר לוח שחמט על המסך (יש לכם את החופש לקבוע כיצד יראה הלוח) כאשר בכל אחת מהמשבצות אשר מופיעות ברשימה lst יופיע מספר בין 1 ל- N ששווה למיקומה ברשימה. בדוגמא לעיל $N=4$ - בתא A4 יופיע המספר 1, בתא H5 יופיע המספר 2 וכן הלאה.

סעיף 3

נתונות ההגדרות הבאות עבור עץ אשר מתאר את כל מסלולי התנועה האפשריים של פרש החל ממיקום התחלתי שנתון בשורה (עליכם למצוא בעצמכם מה יש לרשום במקום שבו רשומות 3 נקודות על-מנת שההגדרות יעברו קומפילציה. נושא זה קרוי forward declaration):

...

```
typedef struct _treeNodeList {
    struct _treeNodeListCell *head;
    struct _treeNodeListCell *tail;
} treeNodeList;
```

```

typedef struct _treeNode{
    chessPos    position;
    treeNodeList next_possible_positions; // רשימת מיקומים
} treeNode;

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

typedef struct _pathTree {
    treeNode    *root;
} pathTree;

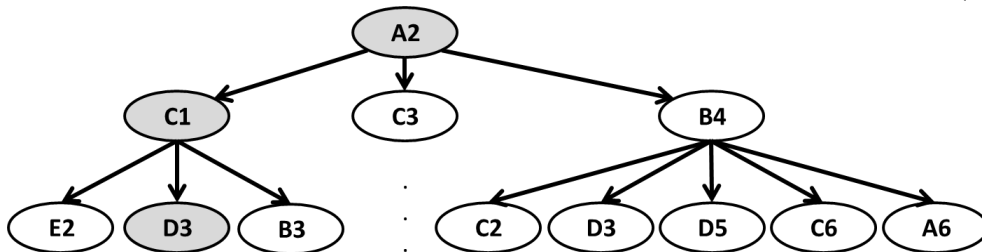
```

צומת כלשהו X בעץ מייצג משבצת בלוח שחמט. ילדיו של הצומת X הם כל המיקומים האפשריים אליהם יכול להמשיך הפרש בצעד אחד מ-X. הילדים נתונים ברשימה המקושרת שראשה נתון ב-`next_possible_positions`.

כדי להימנע ממצב שבו פרש הולך במעגלים, אסור במסלול מסוים לחזור אל משבצת שהופיעה כבר במסלול, כלומר, אסור שרשימת הילדים של X תכיל מיקומים אשר מופיעים כבר במסלול העובר בין השורש ל-X (לדוגמא, בכל צומת שאינו השורש, תמיד אחד מהמיקומים האפשריים שאליו יכול לזוז הפרש הוא צומת האב (לחזור על עקבותיו). לכן אסור, בין השאר, שצומת זה יופיע ברשימת מיקומי ההמשך שמהווים את ילדיו של הצומת).

לדוגמא:

אם המיקום ההתחלתי של הפרש הוא A2 אזי תיאור חלקי של מסלולים באורך 2 (3 הרמות העליונות של העץ) ייראה כדלקמן:



לנוחיותכם, המסלולים מתוארים גם בלוח הבא כאשר מיקומי המסלול המודגש בעץ מודגשים גם בלוח. תוכן המשבצת מורכב ממספר שהוא הרמה בעץ ואות שמתארת את מספר האפשרויות (האפשרויות ממוספרות משמאל לימין).

	1	2	3	4	5	6	7	8
A		1				2c3e		
B			2a3c	2c				
C	2a	2c3a	2b			2c3d		
D			2a3b/ 2c3b		2c3c			
E		2a3a						
F								
G								
H								

שימו לב שמגבלת החזרה שמתוארת לעיל מונעת מעבר של פרש העומד ב-D5 אל B4 היות וביקרנו בו לפני שהגענו ל-D5. יחד עם זאת, ייתכן שמשבצת תשתתף ביותר ממסלול אחד – לדוגמא D3. יש לכתוב את הפונקציה:

```
pathTree findAllPossibleKnightPaths( chessPos *startingPosition)
```

הפונקציה בונה עץ עם כל מסלולי הפרש האפשריים אשר מתחילים במשבצת שמתקבלת כפרמטר. בכל צומת X הפונקציה מחפשת את כל הצמתים האפשריים שאליהם יכול הפרש להמשיך תוך הימנעות מהליכה במעגלים. את כל צמתי ההמשך האפשריים יש לשמור ברשימה ולאחסנם בצומת X. יש למצוא את המשכי המסלולים מכל אחד מילדים. אם לא ניתן להמשיך לאף צומת מ-X אזי X נמצא בסוף אחד מהמסלולים אשר החלו בשורש והוא יהיה עלה בעץ המוחזר.

שימו לב: יש לעשות שימוש בפונקציה אשר יצרתם בסעיף 1.

סעיף 4

בסעיף זה עליכם לכתוב את הפונקציה:

```
chessPosList *findKnightPathCoveringAllBoard( pathTree *path_tree)
```

הפונקציה מוצאת מסלול שמכסה את כל משבצות הלוח החל משורש עץ המסלולים שהתקבל כפרמטר. המסלול ייוצג כרשימה מקושרת אשר התא הראשון שבה יכיל את המיקום שבשורש עץ המסלולים ויתר המיקומים ברשימה יהיו התאים הבאים במסלול לפי סדרם. במידה ואין מסלול אשר עובר דרך כל משבצות הלוח, הפונקציה תחזיר NULL.

פונקציית main

עליכם להגדיר לוח בגודל 5*5 (הדוגמאות הן על לוח סטנדרטי של 8*8 אולם אתם מתבקשים לעבור אך ורק עם לוח 5*5).

פונקציית ה-main תקלוט מהמשתמש מיקום על הלוח, יש לוודא שהוקלד מיקום חוקי. בהמשך, ייבנה עץ של כל המסלולים האפשריים שמתחילים מהמיקום שהוזן, ואז יש למצוא מסלול פרש אשר מכסה את כל הלוח ולהציג אותו באמצעות הפונקציה מסעיף 2. לצורך אחידות בטסטים הקפידו על הסדר הבא במציאת מסלול פרש:

יש להתחיל מהמהלך "שני צעדים למעלה, צעד אחד ימינה" ואז עם כיוון השעון, וכך בכל משבצת במקרה שיש מספר מסלולי פרש אפשריים ממנה.

הנחיות כלליות:

יש להקפיד על חלוקה ל-5 קבצים לפחות.
יש להקפיד על יעילות וחסכון בזמן ריצה וזיכרון.
יש להקפיד שגודל פונקציה לא יחרוג ממסך אחד.
יש לפנות זיכרון שהוקצה דינמית ואשר אין בו צורך יותר.
יש להשתמש ב-#define היכן שנחוץ.

ב ה צ ל ח ה