מבוא למדעי המחשב, 10006 סמסטר ב' – תש"פ

תרגיל הגשה מס' 2

משתנים, קבועים, משפטי תנאי, לולאות
מערכים, מטריצות ופונקציות

הנחיות כלליות

1. יש להגיש ביחידים בלבד
2. ההגשה תתבצע דרך ה-moodle בלבד
3. תאריך ההגשה כפי שמפורסם באתר הינו סופי, לא ניתן להגיש לאחר תאריך זה
4. יש להגיש קובץ ZIP בשם HW2_<ID>.ZIP , כאשר <ID> הינו ת"ז של המגיש, לדוגמא: לדוגמא HW2_005313286.zip
5. בתוך קובץ ZIP יהיו 2 קבצי JAVA
6. כל קובץ יכיל הערה בראש הקובץ עם שם הסטודנט, מס' ת"ז ומספר התרגיל
7. כל הקבצים יוגשו מוכנים להרצה וללא שגיאות קומפילציה
8. בידקו את הקוד באופן אוטומטי על יותר מדוגמא אחת , חישבו על מקרי קצה ונתחו אותם

הנחיות לקידוד

1. יש להקפיד על קוד פשוט, ברור, קריא, מסודר ויעיל
2. יש להשתמש באדינטציה (הזחה), ולרשום הערות
3. יש להקפיד על שימוש בקבועים ושמות משמעותיים למשתנים ולמחלקות
4. יש להשתמש בשמות לפי כללי Java, lower camelCase עבור משתנים ו UPPER_CASE עבור קבועים
5. יש להשתמש בקבועים במידת הצורך
6. במטלה זו ניתן להשתמש בחומר שנלמד בלבד – משתנים, קבועים, משפטי תנאי ולולאות, מערכים, מטריצות ופונקציות - <u>אין להשתמש בנושאים מתקדמים יותר</u>
7. <u>כלל אצבע עבור פונקציות, אורך פונק׳ צריך להיות לא יותר מ-20 שורות קוד</u>

<u>*כל ההנחיות מעלה הינן חובה</u>

תרגיל 1

מסלול מוקשים:

בתרגיל זה נחפש **מסלול** יציאה ממבוך מוקשים

מסלול במטריצה הוא רצף של איברים צמודים ,המתחיל על המסגרת החיצונית של המטריצה ומסתיים בנקודה אחרת גם כן על המסגרת החיצונית של המטריצה

התווים שבמשחק הם:

1. כיוונים (חצים): '<','>','^','V'
   למשל אם בתא במסלול יש את התו v משמע התו הבא במסלול צריך להיות מתחתיו ,ואם התו במסלול הוא > התו הבא במסלול צריך להיות משמאלו
2. צומת: '+'
   מצומת ניתן לצאת בכל כיוון חוץ מהכיוון ממנו הגענו ובתנאי שהכיוון איננו חסום ע"י מוקש או ע"י גבולות המטריצה, מצומת תמיד יש יציאה והיא תמיד אחת, אין 2 צמתים ברצף
3. מוקש: '*'
   חוסם את היציאות הלא נכונות מהצמתים ומוריד חיים
4. דשא: 'O'
   מסמל שטח ריק, אין לנוע בו

הנחיות כלליות:

1. קלוט את גודל השורות והעמודות (מינימום 5 מקסימום 15, לא בהכרח מטריצה ריבועית)
2. קלוט מטריצה המכילה מסלולים מלאים ו/או חלקיים (מסלול חלקי הוא מסלול המורכב מהתווים המתוארים מעלה אך אין לו יציאה ו/או כניסה)
3. אין צורך בבדיקת קלט, הנח שהתווים תקינים
4. קלוט כמות חיים התחלתית
5. שלב מקדים - זהה את המסלול המלא התקין <u>הארוך ביותר</u> שקיים על המטריצה:
   כלומר במידה ויש יותר מכניסה אחת והראשונה לא הובילה למוצא יש לבדוק גם את הכניסות הבאות עד להצלחה או כישלון בכולן
6. שלב מקדים - במידה ולא קיים מסלול הצג הודעות מתאימות
7. לאחר מציאת המסלול בשלב המקדים יש לממש את המשחק לפי החוקים מטה
8. בסיום המשחק יש להדפיס את כל המסלול עד לנקודה האחרונה אליה הגעתם (אין צורך להדפיס מסלול בשלב המקדים של בדיקת המסלולים)
9. לעזרתכם מצורף פתרון של בעיה דומה קלה יותר, ניתן להתבסס עליו במידה ותרצו, תיאור הבעיה ראו בנספח א'

חוקים:

1. כיוון התנועה יכול להיות כל כיוון
2. נקודת הכניסה יכולה להיות על כל אחת מצלעות המטריצה
   - כניסה מהצלע העליונה תמיד תתחיל ב v
   - כניסה מהצלע התחתונה תמיד תתחיל ב ^
   - כניסה מהצלע הימנית תמיד תתחיל ב >
   - כניסה מהצלע השמאלית תמיד תתחיל ב <
3. בהגעה לצומת יש לסרוק את היציאות האפשריות עם כיוון השעון באמצעות פונקציה lookAhead, הצצה קדימה, אם נתקלת במוקש בעת ההצצה יש להוריד חיים. יש לזכור מאיפה הגענו ולא לבדוק קואורדינטה זו (למשל, אם הגענו מקואורדינטה (3,2) ל (3,3), לא נבדוק את (3,2))
4. יש להניח כי תמיד קיימת יציאה אחת תקינה מצומת (^,V,<,>)
5. בהגעה לצומת ללא חיים, או אם החיים נגמרו תוך כדי סריקת הכיוונים, אין אפשרות לסריקה נוספת באמצעות lookAhead, אלא יש להציג למשתמש בחירה:
   "האם אתה מוכן להסתכן?"
   1. במידה וכן –יש לבקש מהמשתמש להזין ידנית תו לכיוון היציאה מהצומת (<>^V),
   2. במידה ולא – יש לעצור את המשחק, להדפיס הודעה מתאימה ("המשתמש לא מוכן לקחת סיכון") ולהדפיס את המסלול
6. היתקלות במוקש ללא חיים תוביל לסוף המשחק, יש להציג הודעה מתאימה ולהדפיס את המסלול
7. על כל צעד יש להדפיס בקונסול פלט המתאר את המהלך ובנוסף לשמור את הצעד לטובת ההדפסה בסוף המשחק
8. היתקלות בגבול המטריצה בעת סריקת צומת תוביל לסריקה הבאה בתור
9. יש לזהות תנועה מעגלית ולעצור את התוכנית (כל אסטרטגיה לזיהוי תתקבל)
10. יש לתת דגש על יעילות
11. יש להשתמש בפונקציות עזר ככל שיידרש
12. יש להדפיס המסלול בצורה נאותה לקריאה, דוגמא נניח נק' כניסה (0,0) ונק' יציאה (2, 2):

.a (0, 0) -> (1, 0) -> (1, 1) -> (1, 2) -> (2, 2)

להלן דוגמא למטריצה ולפלט התוכנית:

o	v	o	^	<	<
o	v	v	*	^	^
*	+	*	o	*	+
o	v	v	*	*	^
o	>	>	>	>	^

```
Starting at (0,1) with 2 lives...
Moving down
Moving down
I have reached a junction,
I'm going to check the area...
Looking to (2,2)... BOOOOOM!!!
  One life is gone, 1 more left...
Looking to (3,1) CLEAR!!!
  Let's go...
Moving down
Moving right
Moving right
Moving right
Moving right
Moving up
Moving up
I have reached a junction,
I'm going to check the area...
Looking to (1,5) CLEAR!!!
  Let's go...
Moving up
Moving left
Moving left
Moving up
Thanks god, we are out, good job…
Exit on (-1,3)

(0,1)->(1,1)->(2,1)->(3,1)->(4,1)->
(4,2)->(4,3)->(4,4)->(4,5)->(3,5)->
(2,5)->(1,5)->(0,5)->(0,4)->(0,3)
```

דוגמא 2 – במסלול הארוך ביותר הגיע לצומת ללא חיים ובחר לא להמשיך

```
o v o ^ < <
o v v * ^ ^
* + * o * +
o v v * * ^
o v o > > o
```

```
Starting at (0,1) with 0 lives...
Moving down
Moving down
I have reached a junction,
Will you take a chance to guess a direction?
no
You lose because you are a CHIKEN!

(0,1)->(1,1)->(2,1)
```

דוגמא 3: - במסלול הארוך ביות הגיע לצומת ללא חיים, בחר להמשיך ועלה על מוקש

```
o v o ^ < <
o v v * ^ ^
* + * o * +
o v v * * ^
o v o > > o
```

```
Starting at (0,1) with 0 lives...
Moving down
Moving down
I have reached a junction,
Will you take a chance to guess a direction?
yes
Which direction you want to continue on?
>
BOOOM!!! Game Over

(0,1)->(1,1)->(2,1)->(2,2)
```

דוגמא 4: - בצהוב מסלול מעגלי שיש לזהות ולפסול בשלב המקדים, בכחול המסלול היחיד התקין, תווי החצי הנוספים לא קשורים לאף מסלול מכיוון שאין להם כניסה

```
o v o ^ < <
o v v * ^ ^
* v * o v <
o v v * v ^
o > > > > ^
```

תרגיל 2

בשאלה זו נממש אלגוריתם לפתרון Murble Puzzle. בבעיה זו ישנו מערך תווים שאורכו 2*size+1 כאשר יש במערך size תווים עם התו 'X' ו- size תווים עם התו '0' כאשר הם מסודרים לסירוגין, והאיבר האחרון הוא ריק.

דוגמה עבור size=3:

```
0 ! X ! 0 ! X ! 0 ! X !   !
```

דוגמה עבור size=4:

```
0 ! X ! 0 ! X ! 0 ! X ! 0 ! X !   !
```

עליכם לכתוב תוכנית המבקשת מהמשתמש את size ומסדרת את איברי המערך כך שכל העיגולים יהיו בצד שמאל וכל האיקסים יהיו בצד ימין.

דוגמה: עבור size=3 המערך יסודר להיות

```
0 ! 0 ! 0 !   ! X ! X ! X !
```

דוגמה: עבור size=4 המערך יסודר להיות

```
0 ! 0 ! 0 ! 0 !   ! X ! X ! X ! X !
```

על מנת לבצע סידור זה, יש לציית לחוקים הבאים:

בכל סיבוב יוחלף המיקום הריק או עם אחד האיברים הסמוכים לו (פעולה זו תקרא shift), או עם איבר המרוחק ממנו מקום אחד, בתנאי שבאמצע בינהם יש תו אחר (פעולה זו תקרא jump).

פלט התוכנית יציג את מצב המערך לאחר כל סיבוב, תוך ציון מהי הפעולה שבוצעה (S יסמן shift, J עבור jump, L עבור left ו- R עבור right).

דוגמת פלט עבור size=3:

```
0 ! X ! 0 ! X ! 0 ! X !   !     SR
0 ! X ! 0 ! X ! 0 !   ! X !     JR
0 ! X ! 0 !   ! 0 ! X ! X !     JR
0 !   ! 0 ! X ! 0 ! X ! X !     SL
0 ! 0 !   ! X ! 0 ! X ! X !     JL
0 ! 0 ! 0 ! X !   ! X ! X !     SR
0 ! 0 ! 0 !   ! X ! X ! X !
```

<u>הדרכה:</u>

ראשית מיצאו את חוקיות ההזזות באמצעות השלמת הטבלאה הבאה עבור לוח עם size=3. לצורך הדוגמה מולאו רק שתי השורות הראשונות:

תוכן המערך						
0	1	2	3	4	5	6
0	x	0	x	0	x	
0	x	0	x	0		x
0	x	0		0	x	x

סוג התזוזה			
Move / Jump	מתא	לתא	כיוון
Move	5	6	R
Jump	3	5	R

מלאו באופן דומה טבלה עבור לוח עם size=6.

חישבו: מהי חוקיות ההזזה? לאיזה כיוון תמיד זז x ולאיזה כיוון תמיד זז 0?מה קורה כאשר יש רצף של תאים זהים?

לאחר שתענו על שאלות אלו ניתן לפנות ולפתור את הקוד:

כתבו תוכנית הממשת משחק זה. הקפידו על חלוקה נכונה לפונקציות ועל מודולריות.

נספח א:

מסלול במטריצה הוא רצף של איברים צמודים משמאל או מלמטה, המתחיל באיבר הימני העליון של המטריצה ומסתיים באיבר כלשהו בשורה התחתונה. כיוון התנועה ברצף זה הינו שמאלה או מטה בלבד. רצף התווים מכיל את התווים '|' ו/או '-' בלבד. התווים הללו מעידים על מיקום התו הבא ברצף.

למשל אם בתא במסלול יש את התו '|' משמע התו הבא במסלול צריך להיות מתחתיו, ואם התו במסלול הוא '-' התו הבא במסלול צריך להיות משמאלו.

התו בשורה התחתונה חייב להיות התו '|'.

הגרל את מימדי מטריצה ריבועית בטווח 4-10 וקלוט לתוכה תווים מהמשתמש והצג את המטריצה.
בדוק האם קיים מסלול של קווים מהפינה הימנית העליונה ועד לאיבר כלשהוא בשורה התחתונה והצג הודעה מתאימה.

<u>דוגמאות</u>:

עבור המטריצות הבאות יוצג true מאחר ויש מסלול עפ"י ההגדרה הנ"ל.

```
a a a a a ¦
a a a a a ¦
a a a ¦ - -
a a ¦ - a a
a ¦ - a a a
```

```
a a a a a ¦
a a a a a ¦
a a a ¦ - -
a a ¦ - a a
a a ¦ a a a
```

עבור המטריצה הבאה יוצג false מאחר שאין מסלול, כי התו בשורה התחתונה הוא '-' ולא '|'.

```
a a a a a ¦
a a a a a ¦
a a a ¦ - -
a a ¦ - a a
a a - a a a
```

עבור המטריצה הבאה יוצג false מאחר שאין מסלול, כי באיבר הימני ביותר בשורה השלישית יש '|', ואז האיבר הבא במסלול היה צריך להיות מתחתיו ולא לידו.

```
a a a a a ¦
a a a a a ¦
a a a ¦ - ¦
a a ¦ - a a
a a ¦ a a a
```

בכל הדוגמאות האלה, במקום התו 'a' יכול להופיע כל תו אחר, כולל התווים '|' ו- '-' כאשר אינם חלק מהמסלול.