

## **מבוא למדעי המחשב מ'ח' (234114/7), סמסטר אביב 2015**

---

### **תרגיל בית 3 - טיוטה**

מועד אחרון להגשה: יום רביעי 23/12/2015 עד שעה 23:00

---

המתרגל האחראי על תרגיל זה: ינון חורש

משרד: טאוב 403

E-mail: ynon980@cs.technion.ac.il

שעות קבלה מיוחדות לשאלות על התרגיל (בנוסף על שעת הקבלה הרגילה) :

**יום ראשון, 16:30 – 17:30**

אם שעות אלו אינן נוחות לכם ניתן לתאם פגישה בשעות אחרות.

**תרגיל זה הינו בגרסת טיוטה. אם יש הערות/הארות על התרגיל אתם מוזמנים לפנות למתרגל האחראי על התרגיל (ינון) או למתרגל האחראי על הקורס (נחשון).**

## הנחיות לתכנון וכתיבת קוד: חשוב לקרוא לפני התרגיל(!)

בתרגיל זה המטרה היא לתרגל אתכם בפירוק בעיה לגורמים קלים לתכנות (ותכנון). לצורך כך, ישנן כמה מגבלות על כתיבת הקוד. **שימו לב, התוכנית תיבדק באופן ידני, ויורדו נקודות על חריגה מהכללים.** בנוסף, הדגש בתרגיל זה הוא על תכנון נכון של הקוד, ובהתאם לכך יינתן לבדיקה הידנית משקל גבוה יותר מהרגיל: 67% מהציון.

1. אורך כל פונקציה לא יעלה על 16 שורות קוד (ראו הגדרות מדויקות בהמשך הדף). חשוב: עמידה בתנאי זה דורשת תכנון מוקדם של הקוד!
    - a. בונס 5 נקודות אם הקוד יעמוד בכל הדרישות, ואורך הפונקציה המקסימלית  $\leq 13$ .
    2. חתב כל שורה (כולל הערות והזחות) לא יעלה על 75 תווים. ניתן לבדוק אורך של שורה בקודבלוקס (ראו תמונה מצורפת עם התרגיל). אם השורות ארוכות הן יגלשו בהדפסה, מה שיקשה על בדיקת התרגיל שלכם (ויגרוור הורדת ניקוד).
    3. לפני כל פונקציה, יש לכתוב בהערה (בקצרה) מה הפונקציה עושה
      - a. ההערה צריכה להסביר מה הפונקציה עושה ולא כיצד היא עושה זאת. לדוגמה: "הפונקציה מכניסה את  $x$  ו  $y$  למערך moves אינו הסבר, לעומת "הפונקציה שומרת את המהלך האחרון במערך moves" הוא כן הסבר.או: "הפונקציה מקדמת את  $num$  אם  $isMove$  חיובי" אינו הסבר, לעומת "אם בוצע מהלך חוקי, הפונקציה מקדמת את מספר המהלכים החוקיים" (או בקצרה, "הפונקציה סופרת את מספר המהלכים החוקיים").
    - b. ההערה צריכה להיות באנגלית. לצערנו עברית מודפסת כג"ברישי.
    - c. ההערה צריכה להופיע לפני המימוש של הפונקציה ולא ההצהרה שלה.
  4. חובה לתת שמות משמעותיים לפונקציות.
    - a. השם צריך לשקף את מה שכתבתם בהערה (אך בתמצות).
- בנוסף, יש להקפיד על הכללים הרגילים לגבי כתיבת קוד נכון:
5. חובה לתת שמות משמעותיים למשתנים שאתם מגדירים.
  6. חובה להשתמש בהזחות תקינות כפי שנלמדו בתרגולים.
  7. חובה להשתמש ב `define` במקומות המתאימים, כפי שנלמדו בתרגולים.
  8. אסור להשתמש במשתנים גלובאליים או סטאטיים.
  9. אסור לשכפל קוד שלא לצורך.

### הגדרות לגבי אורך פונקציה

1. מגבלת השורות תקפה לגבי כל פונקציה בקוד, **כולל ה-main**. יש לרשום את האורך בהערה לפני הפונקציה כדי להקל על הבדיקה.
2. אסור לכתוב 2 פקודות באותה שורה (למשל `x=0; counter++`). לעומת זאת, אפשר להגדיר כמה משתנים באותה שורה ולאתחל אותם.  
מקרה יוצא דופן הוא הפקודות `break – case` בבליק של `switch` בלבד, אותן ניתן לכתוב באותה שורה.
3. שורה המכילה הערות בלבד, כותרת הפונקציה, סוגר `{, }` אחד בלבד, או מילת `else` בלבד אינה נחשבת לספירת אורך הפונקציה.
4. כל שורה אחרת נחשבת לאורך הפונקציה, כולל: הגדרות משתנים, פקודות `if`, פקודת `return` וכל פקודה אחרת שאינה מופיעה ב 3.
5. שורה ארוכה שנשברה לשתיים (כדי שרוחב השורה יהיה קטן מ 75 תווים) נחשבת כ 2 שורות.

## **בקרת עמיתים (בנוס):**

בקורס מבוא למדעי המחשב תלמדו את הבסיס לתכנות. בסיס זה ישמש אתכם בהמשך התואר, וכן בעבודה בחברות הייטק. בעוד שבקורסים סטנדרטים במדעי המחשב המיקוד הוא על כתיבת קוד, בתעשייה תדרשו בעיקר "לתקן" קוד קיים – להוסיף יכולות או לתקן באגים קיימים. מעבר ליכולת לכתוב קוד, נדרש גם (ובעיקר) לקרוא קוד קיים, ולהבין אותו לעומק. בנוסף, מכיוון שמתכנתים מתעסקים בעיקר בקריאת קוד קיים, הכתיבה של קוד קריא וברור הפכה ליכולת חשובה מאוד.

בתרגיל זה אפשר להשתתף בבקרת עמיתים עבור בנוס של 15-18 נקודות. סטודנט המשתתף בסדנה יתבקש להעריך קוד של 2 סטודנטים, והקוד שהגיש יוערך ע"י 2 סטודנטים אחרים. הסטודנטים יספקו לעמיתיהם משוב וציון על קוד שקראו, ויקבלו ציון ומשוב מהסטודנטים שהעריכו את הקוד שלהם. סטודנט שישתתף בסדנה וימלא אחרי כל הוראות הסדנה יקבל בנוס של 15 נקודות. בנוסף יהיה בנוס של עד 3 נקודות שתלוי בציון שתקבלו ע"י סטודנטים אחרים. **סטודנט שלא מעוניין להשתתף בסדנה יכול לקפוץ להמשך התרגיל בעמוד הבא.**

**הגשה:** בנוסף להגשת התרגיל דרך אתר הקורס (באופן הרגיל) יש להגיש את התרגיל גם דרך מודל: <http://moodle.technion.ac.il>. למודל יש להגיש רק את הקובץ hw3q1.c. הקובץ חייב להיות אנונימי ולא להכיל את שמכם או פרט מזהה אחר. הסיסמא הראשונית למודל היא הת"ז שלכם. גם אם הגשתם למודל יש להגיש לאתר הקורס הרגיל. (כמובן שאם אתם לא מעוניינים להשתתף בסדנה אין צורך להגיש למודל).

**הערכה:** ההערכה תתחיל לאחר סיום תאריך ההגשה של התרגיל. **ניתן למצוא טופס הערכה לדוגמה ביחד עם תרגיל זה. מומלץ לעבור על הטופס לפני שאתם מתחילים לכתוב קוד.**

לכל שאלה בקשר לבקרת עמיתים אתם מוזמנים לפנות לנחשון כהן (המתרגל האחראי) באימייל [ncohen@cs.technion.ac.il](mailto:ncohen@cs.technion.ac.il) או שעת קבלה.

## צוללות (Battleships)

בתרגיל זה נממש את המשחק צוללות.

בקישור הבא ניתן לראות המחשה:

[https://he.wikipedia.org/wiki/%D7%A6%D7%95%D7%9C%D7%9C%D7%95%D7%AA\\_%D7%9E%D7%A9%D7%97%D7%A7](https://he.wikipedia.org/wiki/%D7%A6%D7%95%D7%9C%D7%9C%D7%95%D7%AA_%D7%9E%D7%A9%D7%97%D7%A7)

### החוקים:

במשחק שני שחקנים (במקרה של התרגיל זה מחשב מול המשתמש). לכל משתמש שני לוחות. לוח אחד למיקום הצוללות שלו ולוח שני למעקב אחר צוללות היריב. כל לוח מורכב מ – 10 עמודות ו – 10 שורות.

על כל משתמש למקם בתחילת המשחק 5 צוללות (הערה: מספר זה יכול להשתנות! שימו לב שאתם כותבים קוד כך שתוכלו לעדכן שינויים בקלות). במשחק ישנם 5 צוללות לכל שחקן והצוללות הינן בגודל: 2,3,3,4,5

כל שחקן בתורו, בוחר באיזה מקום לתקוף את לוח היריב. לוח המעקב אחר היריב יתמלא באינפורמציה הנכונה – אם נמצאה שם צוללת אז יסומן V אחרת יסומן X. צוללת טבעה אם תקפו את כולה. המשחק מסתיים כששחקן הטביע את כל צוללות היריב.

בתרגיל זה, המשחק יהיה בין מחשב למשתמש. על מנת שהמחשב יבחר ערכי אתחול ותקיפה (פרטים נוספים בהמשך) תצטרכו להשתמש ברנדומליות:

### רנדומליות:

על מנת להגריל ערכים "אקראיים" עליכם להשתמש בפונקציות `rand_range`, `srand_range` המסופקות לכם בקובץ `rand.c`.

**את פונ' אלו יש להעתיק בשלמותן לקובץ c שהנכם מגישים.**

בתחילת התכנית יש לקרוא פעם אחת לפונקציה `srand_range` עם ה – `seed` שהמשתמש הזין. לאחר מכן, כדי להגריל מספר בטווח `low` עד `high` (כולל) יש לקרוא לפונקציה `rand_range`.

הערה: ב-C קיימות פונ' דומות: `rand()` ו – `srand()` (שמוגדרות ב – `stdlib`). מאחר והמחשב לא באמת יכול להמציא ערכים רנדומליים, הוא בוחר את המספר שהוא מחזיר ע"י חישוב מסובך. הערך ההתחלתי של החישוב מכונה `seed`. אם הערך ההתחלתי זהה בשתי הרצות שונות הערכים ה"אקראיים" יהיו זהים בשתי הרצות. לא נשתמש בפונ' `rand()`, `srand()` מאחר ועבור הרצות במערכות הפעלה שונות לא יוחזרו אותם ערכים "רנדומליים".

**שימו לב שבמימוש הפונ' `rand_range` השתמשנו במשתנה סטטי. אסור לכם להשתמש במשתנים סטטיים למעט מקרה זה.**

# מהלך המשחק

בתחילת המשחק תוצג הודעת פתיחה:

Welcome to Battleship!

התוכנית תבקש מהמשתמש seed:

Please enter seed:

המשתמש יזין seed כקלט עבור srand\_range כפי שהוסבר לעיל.  
לאחר מכן המשתמש יכניס רמת קושי – כל מספר טבעי גדול (ממש) מ-0. **יפורט בהמשך.**

Please enter level:

## אתחול:

המשתמש והמחשב צריכים לבחור מקומות עבור הצוללות שלהם. מיקום צוללת בגודל מסוים מורכב מבחירת **מיקום התחלתי** – כלומר בחירת 2 אינדקסים לתא המתאים בלוח, ובחירת **כיוון**. בחירת הכיוון תיוצג ע"י בחירת שני מספרים כל אחד בתחום 0,1-1 (אך לא שניהם 0), ותיקבע בדומה לכיוון וקטור במערכת צירים דו ממדית, כאשר המספר הראשון הוא הכיוון בציר ה-X, והשני הכיוון בציר ה-Y. לדוגמא עבור בחירת (1,1-) הכיוון יהיה אלכסון שמאלה-מעלה. עבור בחירת (1,0) הכיוון יהיה מעלה.

## אתחול עבור המחשב:

המחשב יבחר את מקומות הצוללות האלו לפי סדר עולה: כלומר קודם את הצוללת בגודל 2, לאחר מכן בגודל 3 וכו'..

עבור כל צוללת התוכנית תבחר ערכים רנדומליים למיקום הצוללות, ולאחר מכן ערכים רנדומליים לכיוון הצוללת, כפי שהוסבר לעיל.

בחירת הערכים הרנדומליים תהיה בתחום הרצוי. כלומר עבור מיקום צוללות יבחרו שני ערכים,  $x, y$  רנדומליים, כל אחד בתחום 0,...,9.

עבור כיוון הצוללת צריך לבחור 2 מספרים,  $dirX, dirY$ , כל אחד בין 1- ל-1.

**במקרה של שגיאה כלשהי בבחירת ערכי הצוללת – יפורט בהמשך - יש לבחור את ערכי הצוללת מהתחלה.**

**שימו לב** שאתם בוחרים את הערכים בדיוק כפי שמתואר. בחירה בסדר שגוי או בתחום לא נכון תכשיל את הטסטים ותגרום להורדת ציון משמעותית.

## אתחול עבור המשתמש:

התוכנית תבקש מהמשתמש קלט עבור מיקום הצוללות בסדר עולה. כלומר, בתחילה תבקש קלט עבור צוללת בגודל 2 לאחר מכן 3 וכו'.. הקלט יהיה מהצורה:  $(dirX, dirY)$  כאשר  $(x, y)$  כאלו  $x, y$  – זוג קואורדינטות המייצג מיקום התחלתי.

$dirX, dirY$  – כיוון הצוללת פי שהוסבר לעיל.

**לפני בקשת קלט עבור כל צוללת יודפס לוח המשחק של המשתמש ולאחר מכן תודפס ההודעה:**

Enter location for Battleship of size \*:

כאשר \* הינו גודל הצוללת הרצוי.

במקרה של שגיאה כלשהי בבחירת ערכי הצוללת – יפורט בהמשך - יש להדפיס את ההודעה:

Error: Incorrect parameters! Please enter location for the Battleship size \* again:

ולבחור את ערכי הצוללת מהתחלה.

התוכנית תבצע את כל הנ"ל עד לקליטת כל הצוללות הרצויות.

### שגיאות בבחירת ערכי הצוללת:

- מיקום הצוללת חורג מגבולות המשחק: צריך לבדוק לא רק את  $x, y$ . ייתכן ולאחר בחירת הכיוון הצוללת תחרוג מגבולות המשחק.
- לפחות אחד מ- $dirX, dirY$  לא בין -1 ל-1.
- אם  $dirX, dirY$  שניהם 0.
- אם הצוללת שנבחרה חופפת לצוללת אחרת שכבר מוקמה, כלומר קיים תא ששייך לצוללת שמוקמה בעבר, והצוללת הנוכחית כוללת גם את תא זה. שימו לב שהמקרה הבא תקין:



שימו לב שלא כל השגיאות הנ"ל רלוונטיות לאתחול לוח המחשב.

## דוגמת הרצה לקליטת צוללת:

(מפאת חיסכון במקום, דוגמאות ההרצה הן עבור לוח בגודל 5\*5 וצוללות בגדלים (2,3,3))

```
Welcome to Battleship!
Please enter seed:
5
Please enter level:
2
4 |
3 |
2 |
1 |
0 |
  - - - - -
    0 1 2 3 4
Enter location for Battleship of size 2:
0,0 1,2
Error: Incorrect parameters! Please enter location for the Battleship size 2 again:
0,4 1,0
4 | * *
3 |
2 |
1 |
0 |
  - - - - -
    0 1 2 3 4
Enter location for Battleship of size 3:
0,2 1,1
4 | * * *
3 |  *
2 |  *
1 |
0 |
  - - - - -
    0 1 2 3 4
Enter location for Battleship of size 3:
4,0 1,0
Error: Incorrect parameters! Please enter location for the Battleship size 3 again:
2,0 1,1
All battleships have been located successfully!
```

## ביצוע תקיפות:

בעוד שהמחשב פחות חכם מכם, הוא מסוגל לירות הרבה יותר מהר! בכל תור המחשב מצליח לתקוף כמה פעמים את צוללותיכם! מספר הפעמים שהמחשב יכול לתקוף אתכם בכל תור הינו כרמת הקושי שנקלטה בתחילת המשחק.

המשחק בתורות כאשר השחקן מתחיל ראשון.

### ביצוע תור המשתמש

בתחילת כל תור של השחקן יודפס מצב המשחק – לוח המעקב של השחקן (בפרט בתחילת המשחק יוצג הלוח "הריק") ולאחר מכן לוח המעקב של המחשב עם מיקום צוללות השחקן.

לוח ריק יהיה למעשה מלא ע"י התו ' ' (רווח).

תא בלוח מעקב שהתגלתה בו צוללת יסומן ע"י 'V' ואילו תא שהתגלה שאין בו צוללת יסומן ע"י 'X'.

חלקי צוללות השחקן שמוצגים על גבי לוח המעקב של המחשב ועדיין לא הותקפו יסומנו ע"י '\*'.

מתחת לכל עמודה ומצד שמאל של כל שורה יוצג מספר העמודה/שורה 0-9.

מתחת להדפסת מצב המשחק, תוצג הודעה לשחקן הנוכחי, שתבקש ממנו להזין את המיקום בלוח היריב שבו הוא רוצה לתקוף.

```
Your following table:
4 |  V
3 |
2 |
1 |
0 |
  - - - - -
    0 1 2 3 4
The computer's following table:
4 | * * *
3 |  *
2 | *      X *
1 |          *
0 |  X *
  - - - - -
    0 1 2 3 4
It's your turn!
Enter coordinates for attack:
```

השחקן יכניס את המיקום שברצונו לתקוף בצורה הבאה: x,y

**במקרה של שגיאה כלשהי בבחירת מיקום התקיפה – יפורט בהמשך - יש להדפיס את ההודעה:**

```
Error: Incorrect parameters! Please enter coordinates for attack again:
```

והתוכנית תבקש קלט מחדש עבור תור זה. הודעה זו תימשך עד לקבלת קלט תקין.



### ביצוע תור המחשב

לאחר שהשחקן יבצע את תורו באופן תקין, המחשב יבצע את תורו בצורה רנדומלית, ולאחר מכן יודפס מצב המשחק החדש. ביצוע מהלכי המחשב יהיה באופן דומה לאתחול לוח המחשב: המחשב יגדיל בצורה רנדומלית 2 מספרים בתחום 0-9 אשר יהוו זוג קואורדינטות עד לקבלת זוג קואורדינטות תקין. קואורדינטות תקינות הן קואורדינטות שלא התבצעה מהן אף אחת מהשגיאות המופיעות להלן (למעשה, השגיאה האפשרית היחידה במהלך של המחשב היא תקיפה במקום שכבר הותקף בעבר. שימו לב שלא תודפס הודעת שגיאה במידה וזהו מהלכו של המחשב). המחשב יגדיל מספר זוגות תקינים  $x,y$  לפי רמת המשחק כפי שנקלטה בתחילת התוכנית.

### **שגיאות אפשריות בבחירת מיקום תקיפה-**

- הזנת ערך החורג מגבולות המשחק
- תקיפה במקום שכבר הותקף בעבר

**התוכנית תודיע במידה וצוללת טבעה:** לאחר כל תור (של המשתמש או של המחשב), התוכנית תדפיס את ההודעה הבאה במידה וצוללת כלשהי הותקפה בשלמותה.

- אם הצוללת של המשתמש טבעה:

Your battleship of size \* has been drowned!

- אם הצוללת של המחשב טבעה:

The computer's battleship of size \* has been drowned!

כאשר \* הינו גודל הצוללת שטבעה.

אם בתור של המחשב הוטבעו כמה צוללות, יש להדפיס את ההודעות לפי סדר ההטבעות.

בסיום המשחק, במקרה של ניצחון, תודפס ההודעה הבאה:

Congrats! You are the winner!

במקרה של הפסד, תודפס ההודעה הבאה:

Game over! You lost...

והתוכנית תסתיים.

## יחס לקלט:

- ניתן להניח שתמיד יתקבל "מספיק קלט", כלומר כל עוד המשחק לא נגמר לא נגיע לסוף הקלט (EOF).
- לא ניתן להניח את נכונות הקלט מבחינה סינטקטית. לדוגמא לא מובטח שעבור בקשה לקלט מהצורה  $i, j$  הקלט אכן יהיה בצורה הזו (למשל מספרים שלא מופרדים בעזרת פסיק, או תווים שאינם מספרים או פסיק). במקרים כאלו תודפס השגיאה:

Error: Invalid input!

והתוכנית תסתיים (שימו לב שאסור להשתמש בפונקציה `exit` מאחר ובאופן כללי לא רצוי להשתמש בפונקציה זו וגם היא נמצאת בספריה `stdlib` שאסורה לשימוש).

## מקרה הרצה לדוגמה: (המשך של דוגמת האתחול)

Your following table:

|   |  |  |  |  |  |
|---|--|--|--|--|--|
| 4 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 0 |  |  |  |  |  |

- - - - -  
0 1 2 3 4

The computer's following table:

|   |  |   |   |   |   |
|---|--|---|---|---|---|
| 4 |  | * | * | * |   |
| 3 |  |   | * |   |   |
| 2 |  | * |   |   | * |
| 1 |  |   |   | * |   |
| 0 |  |   | * |   |   |

- - - - -  
0 1 2 3 4

It's your turn!

Enter coordinates for attack:

1,4

Your following table:

|   |  |  |   |  |  |
|---|--|--|---|--|--|
| 4 |  |  | V |  |  |
| 3 |  |  |   |  |  |
| 2 |  |  |   |  |  |
| 1 |  |  |   |  |  |
| 0 |  |  |   |  |  |

- - - - -  
0 1 2 3 4

The computer's following table:

|   |  |   |   |   |   |
|---|--|---|---|---|---|
| 4 |  | * | * | * |   |
| 3 |  |   | * |   |   |
| 2 |  | * |   | X | * |
| 1 |  |   |   | * |   |
| 0 |  | X | * |   |   |

- - - - -  
0 1 2 3 4

It's your turn!

Enter coordinates for attack:

0,4

Your following table:

|   |  |   |   |  |  |
|---|--|---|---|--|--|
| 4 |  | X | V |  |  |
| 3 |  |   |   |  |  |
| 2 |  |   |   |  |  |
| 1 |  |   |   |  |  |
| 0 |  |   |   |  |  |

- - - - -  
0 1 2 3 4

The computer's following table:

|   |  |   |   |   |   |
|---|--|---|---|---|---|
| 4 |  | V | * | * |   |
| 3 |  |   | * |   | X |
| 2 |  | * |   | X | * |
| 1 |  |   |   | * |   |
| 0 |  | X | * |   |   |

- - - - -  
0 1 2 3 4

It's your turn!

Enter coordinates for attack:

2,4

Your following table:

|   |  |   |   |   |
|---|--|---|---|---|
| 4 |  | X | V | X |
|---|--|---|---|---|

|   |  |  |  |  |
|---|--|--|--|--|
| 3 |  |  |  |  |
|---|--|--|--|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 2 |  |  |  |  |
|---|--|--|--|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 1 |  |  |  |  |
|---|--|--|--|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 0 |  |  |  |  |
|---|--|--|--|--|

- - - - -

0 1 2 3 4

The computer's following table:

|   |  |   |   |   |
|---|--|---|---|---|
| 4 |  | V | * | * |
|---|--|---|---|---|

|   |  |  |   |  |   |
|---|--|--|---|--|---|
| 3 |  |  | * |  | X |
|---|--|--|---|--|---|

|   |  |   |  |   |   |   |
|---|--|---|--|---|---|---|
| 2 |  | * |  | X | X | * |
|---|--|---|--|---|---|---|

|   |  |  |  |  |   |  |
|---|--|--|--|--|---|--|
| 1 |  |  |  |  | V |  |
|---|--|--|--|--|---|--|

|   |  |  |   |   |  |  |
|---|--|--|---|---|--|--|
| 0 |  |  | X | * |  |  |
|---|--|--|---|---|--|--|

- - - - -

0 1 2 3 4

It's your turn!

Enter coordinates for attack:

2,3

The computer's battleship of size 2 has been drowned!

Your following table:

|   |  |   |   |   |
|---|--|---|---|---|
| 4 |  | X | V | X |
|---|--|---|---|---|

|   |  |  |   |  |
|---|--|--|---|--|
| 3 |  |  | V |  |
|---|--|--|---|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 2 |  |  |  |  |
|---|--|--|--|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 1 |  |  |  |  |
|---|--|--|--|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 0 |  |  |  |  |
|---|--|--|--|--|

- - - - -

0 1 2 3 4

The computer's following table:

|   |  |   |   |   |
|---|--|---|---|---|
| 4 |  | V | * | V |
|---|--|---|---|---|

|   |  |  |   |  |   |
|---|--|--|---|--|---|
| 3 |  |  | * |  | X |
|---|--|--|---|--|---|

|   |  |   |  |   |   |   |
|---|--|---|--|---|---|---|
| 2 |  | * |  | X | X | * |
|---|--|---|--|---|---|---|

|   |  |  |  |  |   |  |
|---|--|--|--|--|---|--|
| 1 |  |  |  |  | V |  |
|---|--|--|--|--|---|--|

|   |  |  |   |   |  |   |
|---|--|--|---|---|--|---|
| 0 |  |  | X | * |  | X |
|---|--|--|---|---|--|---|

- - - - -

0 1 2 3 4

It's your turn!

Enter coordinates for attack:

3,1

Your following table:

|   |  |   |   |   |
|---|--|---|---|---|
| 4 |  | X | V | X |
|---|--|---|---|---|

|   |  |  |   |  |
|---|--|--|---|--|
| 3 |  |  | V |  |
|---|--|--|---|--|

|   |  |  |  |  |
|---|--|--|--|--|
| 2 |  |  |  |  |
|---|--|--|--|--|

|   |  |  |  |   |  |
|---|--|--|--|---|--|
| 1 |  |  |  | V |  |
|---|--|--|--|---|--|

|   |  |  |  |  |  |
|---|--|--|--|--|--|
| 0 |  |  |  |  |  |
|---|--|--|--|--|--|

- - - - -

0 1 2 3 4

The computer's following table:

|   |  |   |   |   |
|---|--|---|---|---|
| 4 |  | V | * | V |
|---|--|---|---|---|

|   |  |   |   |  |   |
|---|--|---|---|--|---|
| 3 |  | X | * |  | X |
|---|--|---|---|--|---|

|   |  |   |  |   |   |   |
|---|--|---|--|---|---|---|
| 2 |  | * |  | X | X | * |
|---|--|---|--|---|---|---|

|   |  |  |   |  |   |  |
|---|--|--|---|--|---|--|
| 1 |  |  | X |  | V |  |
|---|--|--|---|--|---|--|

|   |  |  |   |   |  |   |
|---|--|--|---|---|--|---|
| 0 |  |  | X | * |  | X |
|---|--|--|---|---|--|---|

- - - - -

0 1 2 3 4

It's your turn!

Enter coordinates for attack:

4,1

```

Your battleship of size 2 has been drowned!
Your following table:
4 | X V X
3 |      V
2 |
1 |      V V
0 |
  - - - - -
    0 1 2 3 4
The computer's following table:
4 | V V V
3 | X *      X
2 | *      X X *
1 |   X X V
0 |   X *      X
  - - - - -
    0 1 2 3 4
It's your turn!
Enter coordinates for attack:
2,1
The computer's battleship of size 3 has been drowned!
Your following table:
4 | X V X
3 |      V
2 |
1 |      V V V
0 |
  - - - - -
    0 1 2 3 4
The computer's following table:
4 | V V V
3 | X * X      X
2 | *      X X *
1 | X X X V
0 |   X *      X
  - - - - -
    0 1 2 3 4
It's your turn!
Enter coordinates for attack:
0,0
Your following table:
4 | X V X
3 |      V
2 |
1 |      V V V
0 | X
  - - - - -
    0 1 2 3 4
The computer's following table:
4 | V V V X
3 | X * X      X
2 | *      X X *
1 | X X X V X
0 |   X *      X
  - - - - -
    0 1 2 3 4
It's your turn!
Enter coordinates for attack:
3,4
Your following table:

```

```
4 | X V X X
3 |      V
2 |
1 |      V V V
0 | X
```

- - - - -

0 1 2 3 4

The computer's following table:

```
4 | V V V X
3 | X * X X X
2 | * X X X *
1 | X X X V X
0 |  X *   X
```

- - - - -

0 1 2 3 4

It's your turn!

Enter coordinates for attack:

3,1

Error: Incorrect parameters! Please enter coordinates for attack again:

1,0

Your following table:

```
4 | X V X X
3 |      V
2 |
1 |      V V V
0 | X V
```

- - - - -

0 1 2 3 4

The computer's following table:

```
4 | V V V X
3 | X * X X X
2 | * X X X *
1 | X X X V X
0 | X X V   X
```

- - - - -

0 1 2 3 4

It's your turn!

Enter coordinates for attack:

2,0

Your battleship of size 3 has been drowned!

Your following table:

```
4 | X V X X
3 |      V
2 |
1 |      V V V
0 | X V V
```

- - - - -

0 1 2 3 4

The computer's following table:

```
4 | V V V X
3 | X V X X X
2 | V X X X *
1 | X X X V X
0 | X X V   X
```

- - - - -

0 1 2 3 4

It's your turn!

Enter coordinates for attack:

3,0

The computer's battleship of size 3 has been drowned!

Congrats! You are the winner!

## דגשים נוספים:

- עליכם לוודא את נכונות הקוד שלכם באמצעות אתר הבדיקה בכתובת <http://cs12.cs.technion.ac.il/~cs234114> כפי שנדרשתם בתרגילים הקודמים. גם הפעם מסופקים לכם באתר הקורס קבצי הקלט והפלט המצופה לטסטים שבאתר הבדיקה.
- בשונה מהתרגילים הקודמים, הפעם תוכלו למצוא באתר גם "תוכנית לדוגמא" (קובץ הרצה). הפלט שהתוכנית מוציאה הוא הפלט המצופה, ולכן תוכלו להיעזר בה כדי לבדוק את עצמכם על מקרים נוספים מעבר לבדיקות המפורסמות באתר. לדוגמא: תוכלו לכתוב קובץ קלט משלכם, להעביר אותו לתוכנית באמצעות redirection ולהשוות את הפלט המתקבל לפלט המתקבל מהתוכנית שלכם. כמו כן, בכל במקרה שההתנהגות הרצויה אינה ברורה, עליכם לבדוק מה התוכנית לדוגמא עושה.
- עליכם לשים לב להדפיס את הפלט כפי שהוא מופיע במדויק בטסטים ובתוכנית לדוגמא, בדגש על: רווחים (בפרט בתחילת שורה או בסופה), מעברי שורה (בפרט בסוף הקובץ) ואותיות גדולות וקטנות.

**בהצלחה!**