נושאים מתקדמים בתכנות סמ' ב' תשע"ז - תרגיל

גירסה 1.02 – תאריך הגירסה: 30/03/2017 (תאריך ההגשה מפורסם באתר הקורס)

1. **הסבר כללי על התרגיל:**
במהלך הקורס נכתוב משחק "צוללות".
המשחק ייכתב על-פני ארבעה תרגילים, כאשר בשלב הראשון נתחיל במשחק פשוט ובהמשך נרחיב את המשחק ואת אפשרויות ההרצה שלו.

2. **סביבת הפיתוח:**
את המשחק נכתוב בווינדוס, בסביבת Visual Studio 2015 הניתנת להורדה בחינם:
https://www.microsoft.com/en-us/download/details.aspx?id=48146
- תוסף מומלץ ל- Visual Studio הוא ++ReSharper C:
https://www.jetbrains.com/resharper-cpp/

3. **תאור המשחק:**
במשחק "צוללות" ישנם שני מתמודדים A ו-B.
לוח הקרב של שני המתמודדים מוכן מראש ונסתר מעיני היריב (כל שחקן מקבל את נתוני הלוח שלו בלבד). מתמודד A מתחיל ראשון.
בכל תור המתמודד בוחר משבצת בלוח של היריב. אם הוא פגע בכלי של היריב הוא מקבל תור נוסף.
אחרת, התור עובר ליריב.
לכל כלי שיט ישנו ניקוד שונה ואורך שונה. על-מנת להטביע כלי שיט יש לפגוע בו בכל הנקודות שמרכיבות אותו.

4. **סיום המשחק:**
המשחק מסתיים כאשר אחד המתמודדים "מחסל" את כל כלי השיט של היריב, והמתמודד מוכרז כמנצח.
- בסיום המשחק מוכרז מנצח וניקוד סיום. ניקוד הסיום ישמש כמדד נוסף להצלחה בתרגילים הבאים.
- התוצאות תישמרנה לאורך כל פעולת התוכנית, כמו גם מאזן הניצחונות והניקוד.

5. **כלי השיט:**
לכל מתמודד ישנם **חמישה** כלי שיט מתוך המבחר הבא:

סוג כלי השיט	אורך (במשבצות)	ניקוד עבור הפלה	אות מייצגת
סירת גומי	1	2	B
ספינת טילים	2	3	P
צוללת	3	7	M
משחתת	4	8	D

- רוחבו של כל כלי הוא משבצת אחת, והם נבדלים אחד מהשני רק באורכם.
- האות המייצגת – האות שמסמנת את כלי השיט בקונפיגורציית לוח המשחק. עבור שחקן A באותיות גדולות ועבור שחקן B באותיות קטנות.

6. **לוח המשחק:**

לוח המשחק הינו בגודל 10 על 10.

להלן דוגמה ללוח משחק (הצבעים הם להמחשה בלבד):

	1	2	3	4	5	6	7	8	9	10
1				B						
2		P						p	p	
3		P								
4						M	M	M		
5	m			d						
6	m			d						b
7	m			d						
8				d						
9	B									
10				b					P	P

בכחול מסומנות ספינות הגומי, בירוק מסומנות ספינות הטילים, בצהוב הצוללות ובאדום המשחתת.

- השורות והעמודות ממוספרות 1-10 **(שימו לב שהספירה מתחילה מ-1 ולא מ-0)** כך שהפינה השמאלית העליונה היא (1,1).
- השורה העליונה (1-10) והעמודה השמאלית (1-10) בציור למעלה אינן חלק מלוח המשחק עצמו אלא הן רק לצורכי המחשה ותצוגה.
- ניתן להניח כלי על לוח הקרב במאוזן או במאונך.
- המגבלה היחידה על מיקום הכלים היא שבכל המשבצות שמסביב לכלי מסויים אין כלי אחר. (לדוגמא: בלוח למעלה, לא ניתן היה למקם כלי משחק שחלק ממנו ימצא במשבצת (7,5), מכיוון שהמשבצת היא משבצת היקפית של הכלי האדום).

תרגיל 1

בתרגיל זה תבנו תכנית לניהול המשחק וכן אלגוריתם פשוט להרצת משחק באמצעות מהלכים מתוך קובץ.

התכנית תקבל כקלט:

1. קובץ עבור לוח הקרב של שני השחקנים שיכלול את פיזור הספינות על הלוח של שני השחקנים.
2. שני קבצים, אחד עבור כל אחד מהשחקנים, שיתארו את מהלך התקיפה של השחקנים.

תיאור הקבצים:

א. <u>לוח הקרב:</u>

א.1. הקובץ הינו קובץ בעל סיומת sboard

א.2. בכל שורה יש 10 אותיות שמייצגות את 10 העמודות בשורה זו, רווחים עבור משבצות ריקות והאותיות B, P, M, D – 5 כלים עבור שחקן A (באותיות גדולות) ו-5 כלים עבור שחקן B (באותיות קטנות). התכנית צריכה לוודא שקובץ הלוח תקין ועומד בכללים ובמידה ולא להוציא הודעה מתאימה מתוך ההודעות הבאות ולסיים את ריצת התכנית:

Wrong size or shape for ship <char> for player A
Wrong size or shape for ship <char> for player B
Too many ships for player A
Too few ships for player A
Too many ships for player B
Too few ships for player B
Adjacent Ships on Board

Comment [AK1]: Added in ver. 1.02
<u>NOTE 1</u>: illegal ships **will <u>NOT</u> be counted** for the "too few" / "too many" validation, this means that user *might* get also a "too few ships" in case of wrong size.
<u>NOTE 2</u>: wrong size error should be displayed once for each type of ship per each player.

א.3. במידה וישנן מספר שגיאות יש לצבור ולהציג את ההודעות השגיאה הרלבנטיות לפי הסדר שלמעלה ובאופן שלא תוצג אותה שורת שגיאה פעמיים גם אם התרחשה פעמיים.

א.4. יש לתמוך בקובץ עם סיום שורה באמצעות '\r\n' וגם בקובץ עם סיום שורה באמצעות '\n' בלבד.

א.5. יש להתעלם משורות מיותרות (לקרוא רק את 10 השורות הראשונות), מתווים מיותרים בשורה (לקרוא רק את 10 התווים הראשונים), משורות חסרות ומתווים חסרים בשורה (להניח רווחים) – כמובן כל עוד ישנם מספיק כלי שיט לכל שחקן על-פני הלוח.

Comment [AK2]: Added in ver. 1.02

א.6. תווים שאינם מוגדרים יחשבו כרווח ולא כשגיאה.

ב. <u>מהלכי התקיפה:</u>

ב.1. הקבצים יהיו בעלי סיומת attack-a ו- attack-b

ב.2. התקיפות יופיעו לפי סדר, תקיפה אחת בכל שורה, כאשר התקיפה הראשונה בשורה הראשונה וכן הלאה, בפורמט: row,column

ב.3. יש לתמוך בקובץ עם סיום שורה באמצעות '\r\n' וגם בקובץ עם סיום שורה באמצעות '\n' בלבד.

ב.4. יש לתמוך בכמות כלשהי של רווחים מסביב לפסיק.

ב.5. במידה וישנה שורה עם קלט לא חוקי (לדוגמה מספר שורה > 10) יש להתעלם ממנה ולעבור להבאה בתור על-מנת לשחק.

ב.6. במידה והגיע תורו של שחקן לשחק והוא מיצה את כל שורות הקובץ שלו, הוא מפסיק לתקוף.

ב.7. במידה והקבצים של שני השחקנים מוצו ולא נרשם ניצחון המשחק ייגמר ויודפס הניקוד של השחקנים בלבד.

בבדיקה נריץ קבצים שלנו על התכנית שלכם ולכן חשוב שהתכנית תדע להתמודד עם קבצים שונים בפורמט שהוגדר ולהוציא פלט מדוייק כפי שמוגדר.

הקלט:

א. עליכם להגיש תוכנית שמתקמפלת לקובץ בינארי (.exe) – Console Application.
ב. התכנית תדע לקבל ב-command-line ארגומנט אחד שייצג את ה-path שבו יש לחפש את שלושת הקבצים בסיומת שתוארו. במידה ולא נמסר פרמטר התכנית תחפש את הקבצים ב-working directory. במידה ולא נמצא אחד הקבצים או יותר יש להוציא הודעה מתאימה מתוך ההודעות הבאות ולסיים את ריצת התכנית:

Wrong path: <path>
Missing board file (*.sboard) looking in path: <path>
Missing attack file for player A (*.attack-a) looking in path: <path>
Missing attack file for player B (*.attack-b) looking in path: <path>

ג. במידה וניתן Path לא תקין / שאינו קיים יש לתת רק את הודעת השגיאה הראשונה.
ד. במידה וחסר יותר מקובץ אחד יש לצבור ולהציג את הודעות השגיאה הרלבנטיות לפי הסדר שלמעלה.
ה. הארגומנט של ה-path יכול להיות Full Path או Relative Path.
ו. בעת קבלת הקבצים יש לוודא **תקינות הקלט של הלוח בלבד**.

עיצוב הקוד ומהלך המשחק:

1. עליכם לממש מחלקה עבור אלגוריתם "משחק מקובץ", שיממש את המחלקה האבסטרקטית IBattleshipGameAlgo וידע לשחק לפי מהלכים מתוך קובץ.
2. יש לנהל משחק בין שחקן A לשחקן B כאשר שחקן A מתחיל.
3. המחלקה האבסטרקטית IBattleshipGameAlgo מכילה את המתודות האבסטרקטיות הבאות:

```
void setBoard(const char** board, int rows, int cols); // called once to notify player on his board

std::pair<int, int> attack(); // ask player for his move

void notifyOnAttackResult(int player, int row, int col, AttackResult result); // notify on last move result
```

4. המשחק מפעיל עבור כל שחקן את הפונ' setBoard ושולח לכל שחקן את לוח המשחק שלו בלבד. השחקן אמור להעתיק אליו את הנתונים אם הוא מעוניין בהם – אסור לו לשמור את הפוינטר שנמסר לו כי הוא לא יודע מה אורך החיים שלו.
5. המשחק קורא לשחקן A לפונ' attack ומקבל את המהלך של שחקן A.
6. לאחר כל מהלך של שחקן המשחק מדווח לשני השחקנים על תוצאות המהלך באמצעות הפונקציה notifyOnAttackResult כאשר הפרמטר הראשון הוא מספר השחקן שביצע את המהלך – 0 ל-A ו-1 ל-B. הפרמטרים col-Irow מייצגים את המהלך שבוצע ו-AttackResult הינו enum עם הערכים: Hit – פגיעה בספינה / Miss – פגיעה במים / Sink – פגיעה אחרונה בספינה שגורמת להטבעתה.
7. לשחקן שביצע פגיעה מוצלחת מוענק תור נוסף. במידה ולא יש להעביר את התור ליריב.
8. לאורך כל המשחק יש לבצע מעקב אחר מצב הלוחות של שני השחקנים.
9. המשחק מסתיים כאשר אחד השחקנים מטביע את כל ספינות היריב – והוא מוכרז כמנצח. במצב זה יש להדפיס למסך את ההודעה הבאה (השורה הראשונה תודפס רק במידה והושג ניצחון):

Player <A or B> won
Points:
Player A: <A points>
Player B: <B points>

שימו לב שבתרגיל 1 יש להריץ משחק בודד ולסיים את התכנית.

Comment [AK3]: The idea of this base class is to implement in next exercise some other Game Algorithms. In the 1st exercise your Game Algorithm is quite naïve – without real thinking – it ignores the info sent to it and react to the "attack" calls with the predefined moves from file.

טיפול בשגיאות שלא תוארו:

ככלל יש לפתור כל שגיאה שניתן להתגבר עליה ולא תוארה לעיל ולהמשיך בריצה באופן מתאים.

במידה וישנה שגיאה שלא ניתן להתגבר עליה יש להדפיס למסך הודעת שגיאה אינפורמטיבית בפורמט הבא:

Error: <whatever_you_like>
e.g.
Error: Moshe Kojak is sick again…

לאחר מכן יש לסגור את התוכנית באלגנטיות (למשל להחזיר 1-) **ובשום אופן <u>לא להשתמש במתודה ()exit</u>**
לא בשגיאות שתוארו, לא בשגיאות שלא תוארו ולא בכל מצב אחר.

מה מגישים:

א. <u>קובץ ZIP בשם</u>:
ex1_<id1>_<id2>_<id3>.zip
לדוגמא: ex1_123456789_987654321.zip (כאן יש רק 2 סטודנטים).

ב. <u>הקובץ יכיל</u>:
   a. **כל** כל קבצי ה- cpp. וה-h. של התכנית כולל הקובץ IBattleshipGameAlgo.h שסופק לכם!
   b. קובץ בשם: "students.txt" ובתוכו היוזר האוניברסיטאי שלכם ותעודת הזהות, לדוגמא:
      itzikkobra 123456789
      heizenberg 987654321
   c. קובץ CMakeLists.txt מה-moodle – עליכם לעדכן בראש הקובץ את המשתנים הבאים בלבד:
      i. יוזר אוניברסיטאי
      ii. ת.ז.
      iii. רשימת **כל** כל קבצי ה- cpp. וה-h. כולל הקובץ IBattleshipGameAlgo.h שסופק לכם!
      CMake היא מערכת בניית פרוייקטים (מזכיר את makefile בלינוקס) שנועדה להקל על תהליך הבדיקה האוטומטי. מלבד לעדכן את הקובץ, אין לכם שימוש בו.
   d. שלושת קבצי הקלט הנדרשים למשחק: קובץ sboard. , קובץ attack-a. וקובץ attack-b.

<u>בונוס 5 נק'</u> (לא חובה)

תמיכה בהדפסת ריצת המשחק במסך קונסול:

- יש למצוא מימוש של הפונ' gotoxy ברשת, כנ"ל של setTextColor
- לקבל ב-command-line פרמטר quiet– שקובע שלא תהיה הדפסה של הריצה למסך במידה ולא מתקבל פרמטר זה אז כברירת מחדל תהיה הדפסה למסך של מהלך הריצה!
- יש להציג את הספינות של שני השחקנים באופן צבעוני, אנימציה של "הפצצה" באמצעות סימן @ או סימן אחר שתבחרו, סימון של פגיעה באמצעות * או סימן אחר שתבחרו
- מותר לסמן את כלי השיט של השחקנים השונים באופן שתבחרו ובלבד שיהיה ברור למי שייך כל כלי
- יש לקבוע זמן המתנה סביר כברירת מחדל בין מהלך למהלך, על מנת לאפשר צפייה בריצת המשחק ולאפשר לקבוע זמן שונה באמצעות פרמטר command-line: <delay in ms> delay-
- יש לאפשר סדר כלשהו של הפרמטרים ב-command-line