הנדסת תוכנה 094129

תרגיל בית 3

תכנות מודולארי, ירושה, פונקציות וירטואליות, רב צורתיות (Polymorphism)

תאריך אחרון להגשה : 14\1\2018 עד השעה 23:55

נושא התרגיל: מימוש משחק סולמות ונחשים

תיאור התוכנה – דומה לתרגיל בית 2, כולל שינויים כפי שיפורטו בהמשך:

סולמות ונחשים הוא משחק לוח למספר שחקנים, הכולל רגלים, קובייה ולוח משבצות. במשחק המקורי יש מאה משבצות, אף כי יצאו גרסאות שונות למשחק עם כמות משבצות שונה.

כל שחקן מקבל רגלי ומטרת המשחק היא להגיע ראשון למשבצת האחרונה. ההתקדמות נעשית לפי מספר הנקודות שמראה הקובייה.

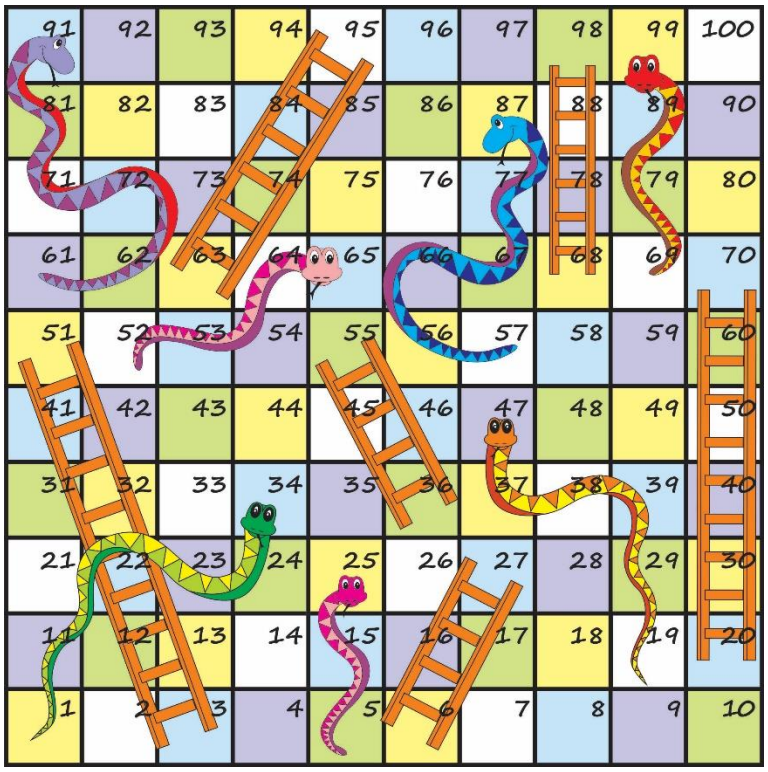
כאשר מגיע שחקן למשבצת שעליה מצויר סולם הוא עולה במעלה הסולם עד למשבצת שראש הסולם מגיע אליו. אם הוא מגיע למשבצת עליה מצויר ראש של נחש הוא יורד עד למשבצת בה נמצא זנבו של הנחש.
הסולמות והנחשים מכניסים עניין למשחק, מכיוון שייתכנו שינויים גדולים במיצובם היחסי של השחקנים.

לוח המשחק:

כפי שצוין לעיל, לוח המשחק מורכב מסולמות ונחשים אשר מטרתם להזיז את השחקנים ולהכניס עניין למשחק.

בתרגיל בית זה, לוח המשחק הינו קבוע ומורכב מ-10 שורות ו-10 עמודות (סה״כ 100 משבצות), כאשר השחקן הראשון שמגיע או חוצה את המשבצת הממוספרת ב-100 הינו הזוכה.

לוח המשחק מוצג בעמוד הבא.



דוגמא למשחק שיש בו שני שחקנים, A ו-B, אשר שניהם מתחילים מנקודת ההתחלה במשבצת (1,1):

נניח ושחקן A מקבל את המספר 2 בקובייה, אז הוא מתקדם למשבצת שמספרה 3, ומשם עולה למשבצת שמספרה 51.

מייד אחריו משחק השחקן B. נניח ושחקן B מקבל את המספר 6 בקובייה, אז הוא מתקדם למשבצת שמספרה 7.

וכך ממשיך המשחק עד שאחד השחקנים מגיע בהצלחה למשבצת מספר 100, או חוצה אותה.

את הלוח של המשחק יש לבנות באמצעות הבונה של המחלקה Template (אשר תפורט בהמשך) והוא יכיל את המטריצה המקודדת בצורה הזו:

```
const int height = 10;
const int width = 10;

int basic_map[height][ width] = {{ 0 , 0 , 48 , 0 , 0 , 21 , 0 , 0 , 0 , 0},
                                 { 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 50},
                                 { 0 , 0 , 0 , 0 , -20 , 0 , 0 , 0 , 0 , 0},
                                 { 0 , 0 , 0 , -33 , 0 , 19 , 0 , 0 , 0 , 0},
                                 { 0 , 0 , 0 , 0 , 0 , 0 , -28 , 0 , 0 , 0},
                                 { 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0},
                                 { 0 , 0 , 32 , 0 , -13 , 0 , 0 , 30 , 0 , 0},
                                 { 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0},
                                 { 0 , 0 , 0 , 0 , 0 , 0 , -30 , 0 , 0 , 0},
                                 {-30 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -30 , 0}};
```

- אפשר להניח כי גודל הלוח לא משתנה בזמן ריצה
- יש לאתחל את פתרון התרגיל עם הלוח הנ"ל
- אפשר לשנות את התוכן של המשבצות של הלוח תוך כדי משחק (מפורט בהמשך)
- בכל משבצת מיוצג מספר, נסמנו כרגע ב-x. במידה ומתקיים ש:
  - $x = 0$: המשבצת ריקה
  - $x < 0$: קיים נחש בתוך המשבצת, אשר מחזיר את השחקן x צעדים אחורה.
  - $x > 0$: קיים סולם בתוך המשבצת, אשר מקדם את השחקן x צעדים קדימה.

מבני הנתונים של התוכנה:

במימוש המחלקות, עליכם להשתמש בטיפוסי הנתונים הבאים (שימו לב שעליכם למקם את המחלקות בקבצי header שונים, בהתאם למבנה הקוד):

```
// A vector of players
typedef vector<Player*> PlayerVec;
typedef vector<Player*>::iterator PlayerVec_it;
typedef vector<Player*>::const_iterator PlayerVec_cit;
// Template types:
typedef int   size_type;
typedef int** grid_type;
// Player status
typedef enum { NON_WINNER = 0, WINNER = 1} status ;
// Player types
typedef enum {
      REGULAR = 0, FAST = 1,
      SLOW = 2, LIMITED = 3,
      STRONG = 4
} playerType ;
```

כחלק מהפתרון, עליכם לממש את המחלקות הבאות, בהתאם לממשק המתואר להלן:

Class name	Class Members (Private)	Public member methods
Coordinate	int x; int y;	Coordinate(int new_x, int new_y); Coordinate(const Coordinate& newCoordinate) ~Coordinate();
Player	Coordinate coordinate; Const std::string name;	Player(const Coordinate& newCoordinate, const std::string& newName); ~Player();
FastPlayer		FastPlayer(const Coordinate& newCoordinate, const string& newName); virtual ~FastPlayer();
SlowPlayer		SlowPlayer(const Coordinate& newCoordinate, const string& newName); virtual ~SlowPlayer();

LimitedPlayer	`const int limit;`	`LimitedPlayer(const Coordinate& newCoordinate, const string& newName, const int limit);` `virtual ~LimitedPlayer();`
StrongPlayer		`StrongPlayer(const Coordinate& newCoordinate, const string& newName);` `virtual ~StrongPlayer();`
PlayerDB	`PlayerVec players;` `Template* template`	`PlayerDB(Template *new_template);` `~PlayerDB();`

Template	`grid_type ppGrid`
	`size_type size_h;`
	`size_type size_w;`
		`Template();`
		`~Template();`

Interface	`PlayerDB* myPlayers;`	`Interface(PlayerDB* pNew_Players);`
		`~Interface(){};`
		`void StartControl(Template* template);`

- כל תכונות המחלקות פרטיים (All class members are private), פרט למחלקה Player בה התכונות הינן protected.
- אין לשכפל מידע קיים ולשמור אותו תחת שם אחר.
- אין לשנות את מימוש המתודות print של המחלקה Player (מימוש של המתודה הנ"ל נמצא בקבצים המצורפים לתרגיל הבית).

הקלט לתכנה:

התכנה קולטת את נתוני הפקודות מערוץ הקלט הסטנדרטי (cin). הקלט מחולק לפקודות, וכל פקודה תופסת שורה אחת בדיוק בקלט. סדר ביצוע הפקודות חייב להיות סדר הופעתן . בקלט.

בהתחלה על התוכנה להציג הודעה זו:

```
Please Enter The Players' Names:
```

ואז לקבל מהמשתמש כקלט את שמות השחקנים וסוגם בצורה זו:

תיאור הפעולה	מבנה שורת הקלט	פרמטרים
קליטת שחקן רגיל	`REGULAR NNN`	NNN הינו שם החקן (string)
קליטת שחקן מהיר	`FAST NNN`	NNN הינו שם החקן (string)
קליטת שחקן איטי	`SLOW NNN`	NNN הינו שם החקן (string)
קליטת שחקן מוגבל	`LIMITED NNN X`	NNN הינו שם החקן (string) X הינו פרמטר הגבלה (int)
קליטת שחקן חזק	`STRONG NNN`	NNN הינו שם החקן (string)

כאשר מסתיימת פעולת הכנסת השמות יש להקליד `Start!` ואז להתחיל את המשחק על ידי הצגת הודעה זו:

```
Let's Start The Game!
```

במהלך המשחק המשתמש מזין את הפקודה הבאה:

```
Play NNN X
```

כאשר Play הינה שם הפקודה אשר יוזמת שלב חדש במשחק.

NNN הינו שדה מסוג String אשר מסמן את השם של השחקן (יכול להכיל יותר מ-3 אותיות), ו-X הינו משתנה מסוג int אשר מייצג את המספר שמראה הקובייה באותו שלב (כמה משבצות השחקן הולך להתקדם).

יש אפשרות במהלך המשחק לשנות את תוכן משבצת מסוימת על ידי הפקודות הבאות:

תיאור הפקודה	**מבנה שורת הפקודה**	**פרמטרים**
שינוי למשבצת ריקה	`AddPath X`	הוספת משבצת ריקה בתוך משבצת מספר X, כאשר X הינו משתנה מטיפוס int
הוספת סולם	`AddLadder X num`	הוספת סולם בתוך משבצת מספר X, המזיז את השחקן num משבצות קדימה. כאשר X ו- num הינם משתנים מטיפוס int
הוספת נחש	`AddSnake X num`	הוספת נחש בתוך משבצת מספר X, המזיז את השחקן num משבצות אחורה. כאשר X ו- num הינם משתנים מטיפוס int.

הפקודות הנ״ל דורסות את תוכן המשבצת הקיים ומעדכנות את לוח המשחק בהתאם.

התוכנה מפסיקה כאשר יש לפחות זוכה אחד במשחק, ומכריזה על השם שלו על ידי הדפסת ההודעה:

```
NNN Wins!
```

כאשר NNN הינו השם של השחקן הזוכה.

התנהגות השחקנים:

בתרגיל זה, נגדיר 4 סוגים של שחקנים אשר מתנהגים באופן שונה:

1. שחקן רגיל: שחקן אשר התנהגותו זהה להתנהגות שהוגדרה בתרגיל בית 2.
2. שחקן מהיר: שחקן אשר לא נופל במגלשות, כלומר ברגע שהוא מגיע למשבצת שיש בה מגלשה הוא עומד בה ולא יורד למטה כפי שעושה שחקן רגיל.
3. שחקן איטי: שחקן אשר לא עולה בסולמות, כלומר ברגע שהוא מגיע למשבצת שיש בה סולם הוא עומד בה ולא עולה למעלה כפי שעושה שחקן רגיל.
4. שחקן מוגבל: שחקן אשר לא יכול לזוז יותר מהשדה limit שלו, כלומר אם הקובייה מראה מספר שהוא גדול יותר מה-Limit שלו הוא יזוז כפי שמוגדר פרמטר ההגבלה שלו, אחרת הוא יזוז לפי מספר הצעדים אשר מראה הקובייה.
5. שחקן חזק: שחקן אשר מתנהג כמו השחקן המהיר, אך בנוסף ברגע שהוא עומד על משבצת המכילה מגלשה הוא דורס את המגלשה והופך אותה למשבצת רגילה.

- בתחילת המשחק, כל השחקנים ממוקמים על המשבצת הראשונה, הממוספרת ב-1.
- השחקנים זזים ימינה בכל שלב במשחק על פי המספר המוצג בפקודת Play.
- כל השחקנים משחקים בתור לפי השמות שהוכנסו לפני תחילת המשחק, ניתן להניח כי הקלט תקין מבחינה זו וכל השחקנים משחקים בסדר הנכון לפי סדר הזנת השמות שלהם לתוכנית.
- בכל שלב במשחק יש להציג את ההודעה הבאה המתארת את מיקום השחקן בלוח המשחק:

```
Player: NNN is at: S
```

כאשר NNN הינו שם השחקן, ו-S הינו מספר מסוג int אשר מייצג את מספר המשבצת בה נמצא השחקן במצב הנוכחי.

- שחקן אשר מגיע למשבצת 100 או חוצה אותה (כלומר יוצא מהגבולות של לוח המשחק) זוכה, ואז התוכנית מכריזה על זה ומסתיימת.
- במידה והשחקן הזוכה חוצה את המשבצת מספר 100, יש להציג הודעה על מיקומו בתוך המשבצת 100 ולא מחוץ לגבולות הלוח.
- אסור להוסיף שחקנים חדשים כאשר המשחק נמצא בתהליך, כלומר הוספת השחקנים מתבצעת אך ורק לפני שהמשחק מתחיל.
- בסוף המשחק יש למחוק את כל השחקנים מהתוכנית בעזרת הדיסטרקטור של מחלקת PlayerDB.

הנחות שמותר לכם להניח לגבי הקלט:

- ניתן להניח כי הקלט מכיל רק פקודות בפורמט הנ"ל.
- ניתן להניח שלא מנסים להוסיף שחקנים חדשים למערכת בתהליך המשחק.
- ניתן להניח כי הקלט תקין מבחינת הסדר שבו השחקנים משחקים.
- ניתן להניח שכל המחרוזות המוזכרות לעיל מכילות רק אותיות באנגלית (גדולות או קטנות) ומספרים. ובפרט, שאינן מכילות רווחים.
- ניתן להניח כי הפרמטר limit של השחקן המוגבל הינו לכל היותר 6.
- בסיום כל פקודת מסוג Play יש לקרוא לפונקצית ההדפסה השייכת לאובייקטים מטיפוס Player:

```
print(const int cell);
```

עם הפרמטרים הרלוונטיים.

- בסיום המשחק יש להדפיס את שם השחקן הזוכה ולהפסיק את התוכנית.

הפלט של המערכת:

בהינתן הלוח למעלה, על המערכת עבור הקלט:

```
REGULAR George
LIMITED Mira 3
Start!
Play George 2
Play Mira 3
AddPath 6
Play George 6
Play Mira 2
AddLadder 9 15
Play George 6
Play Mira 6
AddSnake 27 10
Play George 3
Play Mira 4
Play George 6
```

התכנית תדפיס:

```
Please Enter The Players' Names:
Let's Start The Game!
Player: George is at: 51
Player: Mira is at: 4
Player: George is at: 57
Player: Mira is at: 6
Player: George is at: 95
Player: Mira is at: 24
Player: George is at: 98
Player: Mira is at: 17
Player: George is at: 100
George Wins!
```

דרישות המימוש:

1. המימוש חייב להכיל לפחות את הקבצים הבאים:

א. main.cpp - כפי שניתן על ידי צוות הקורס בקובץ המצורף.
אין לשנות קובץ זה ,והמימוש שלכם חייב להשתמש בפונקציית main המוגדרת בו!

ב. Interface.h – ממשק השליטה במשחק. מותר להוסיף מתודות ו/או תכונות– במקרה הצורך , אין למחוק תכונות/מתודות קיימות

ג. Coordinate.h – ייצוג קואורדינטה במישור. מותר להוסיף מתודות ו/או תכונות– במקרה הצורך ,אין למחוק תכונות/מתודות קיימות

ד. Player.h – מחלקה המייצגת שחקן אחד. כל ההדפסות של מיקומי **כל סוגי השחקנים** יבוצעו אך ורק בעזרת המתודה print(const int cell) של המחלקה הזו. מותר להוסיף מתודות ו/או תכונות – במקרה הצורך ,אין למחוק תכונות/מתודות קיימות

ה. FastPlayer.h – מחלקה המייצגת רובוט מהיר אחד. מותר להוסיף מתודות ו/או תכונות– במקרה הצורך ,אין למחוק תכונות/מתודות קיימות

ו. SlowPlayer.h – מחלקה המייצגת רובוט איטי אחד. מותר להוסיף מתודות ו/או תכונות– במקרה הצורך ,אין למחוק תכונות/מתודות קיימות

ז. LimitedPlayer.h – מחלקה המייצגת רובוט מוגבל אחד. מותר להוסיף מתודות ו/או תכונות– במקרה הצורך ,אין למחוק תכונות/מתודות קיימות

ח. StrongPlayer.h – מחלקה המייצגת רובוט חזק אחד. מותר להוסיף מתודות ו/או תכונות– במקרה הצורך ,אין למחוק תכונות/מתודות קיימות

ה. PlayersDB.h – מכיל ממשק של מודול שמממש את אוסף השחקנים, ז"א שומר את הנתונים במבנה נתונים פנימי וממשש פעולות עליו.

ו. Template.h - מכיל ממשק של מודול שמממש את ממשק לוח המשחק, ז"א שומר את נתוני ופעולות הלוח.

ז. קבצי מימוש של המודולים הנ"ל (קבצי ה-cpp).

ח. קבצי קוד אחרים, במידה ותמצאו זאת לנכון.

2. חובה לתעד את הקוד באנגלית (אין צורך לדאוג לרמת האנגלית) והתיעוד חייב להופיע בתוך הקוד, לפני הפונקציה\טיפוס\משתנה שהתיעוד מתייחס אליהם. נא תעדו ע"פ ההסברים באתר הבא:

http://www.edparrish.net/common/cppdoc.html

3. התיעוד יכיל:

ה. בראש כל מודול תיאור קצר(כמה שורות) של ייעוד המודול והקשר שלו למודולים האחרים.

ו. לפני כל פונקציה תיאור בן שתי שורות של ייעוד הפונקציה, וכן תיאור של כ"א מהפרמטרים שלה (שורה אחת לכל פרמטר) וערך ההחזר שלה. תיעוד הפרמטרים\ערך החזר צריך להכיל הסבר לגבי תוכן המשתנה ,לא הטיפוס שלו.

ז. לפני כל משתנה גלובלי תיאור בשורה של הייעוד שלו .

4. יש לתת שמות בעלי משמעות לכל משתנה! למען הסר ספק:
   ה. השמות i,j,k,l,m עבור משתני לולאה שהם אינדקסים במערך הם כן בעלי משמעות.
   ו. השם it עבור iterator והשם cit עבור const_iterator הם כן בעלי משמעות ,לצורך מילוי הוראה זו.
   ז. פרט למצוין לעיל ,שם שהוא ראשי תיבות לא נחשב לבעל משמעות.
5. אין בתרגיל זה דרישה של יעילות זמן. בפרט, מבנה הנתונים vector בפירוש אינו היעיל ביותר מבחינה זו.
6. עבודה עם (בפרט מעבר על) מיכלים סדרתיים תיעשה אך ורק בעזרת איטרטור המתאים ביותר.
7. הקדישו תשומת לב ומחשבה לאופן בו הפונקציות שלכם מקבלות פרמטרים .השתמשו בהעברה ע״י references במקום העברה by value במקרים בהם הדבר אפשרי וחוסך העתקות. חובה להשתמש ב const - לפי הנלמד בקורס!

הוראות הגשה:

1. התרגיל להגשה בזוגות בלבד.
2. הקוד חייב להיכתב על פי מוסכמות כתיבת הקוד (Coding Conventions) בקורס. קוד שלא עומד במוסכמות לא יזכה במלוא הניקוד.
3. חובה לקמפל את התרגיל בעזרת סביבת Eclipse. תוכנית אשר תצליח להתקמפל בסביבה אחרת ולא בסביבה זו תחשב כקוד שלא עובר קומפילציה.
4. ההגשה חייבת להכיל קובץ ZIP יחיד בלבד (ולא קובץ RAR וכדומה) המכיל:
   - תיקיה בשם code ובה כל קבצי קוד המקור (h/cpp), ללא קבצי הרצה.
5. שם הקובץ חייב להיות hw3_xxxxxxxxxx_yyyyyyyyyy.zip, כאשר xxxxxxxxxx ו-yyyyyyyyyy הם מספרים תעודות הזהות של המגישים, כולל ספרת ביקורת
6. ההגשה היא אלקטרונית בלבד, דרך אתר ה-moodle של הקורס. תרגילים שיוגשו בכל דרך אחרת לא ייבדקו.
7. אין להגיש את אותו הקובץ פעמיים. התרגיל יוגש על ידי אחד מבני הזוג.
8. שימו לב שההגשה תיחסם בדיוק בשעה 23:55. מומלץ להגיש לפחות שעה לפני המועד האחרון.
9. תרגיל בית שלא יוגש על פי הוראות ההגשה - לא ייבדק.

בהצלחה !