AP C++

תרגיל 1

מועד הגשה: 7.11.2018

מטרות התרגיל:

Memory allocation, Constructor, Destructor, Copy constructor, Operator=, Enum ,Const, Vector, makefile

הנחיות הגשה:

- נא לציין בראש כל קובץ שאתם מגישים שם ותעודת זהות לדוגמא:
```
/* Moshe Israel 123456789 */
```
- ההגשה דרך מערכת הsubmit בכתובת:
  http://31.154.73.187/cgi-bin/welcome.cgi
- ניתן להגיש קובץ zip המכיל את הקבצים הנדרשים.
- עליכם להוריד מהאתר את הקבצים ולעבוד עליהם. בהגשה עליכם להגיש **רק חמישה קבצים**:
  makefile Game.h Game.cpp Player.h Player.cpp
- עליכם להגיש קובץ makefile והשם של target הוא a.out
- כמו כן, שימו לב שחלק מהמחלקות ממומשות, או ממומשות חלקית. **מותר לשנות** את תוכן הקבצים שאתם כותבים– העיקר שהתוכנית תעבוד כנדרש.
- אין צורך לשנות שמות קבצים או תוכן של קבצי הCard והmain
- **אלו מכם שעובדים על מערכת הפעלה שהיא לא windows:**
  בקובץ Card.cpp עליכם למחוק את שורה 3 (או להפוך אותה להערה):
  `#define WINDOWS` → `//#define WINDOWS`
  שאר הסטודנטים אין צורך לשנות כלום!
- יש להקפיד על הסכמי משתמשים (coding style):
  - שמות מחלקות מתחילים באות גדולה, שמות משתנים מתחילים באות קטנה, הערות, חלוקה לפונקציות וכו'.
  - יש למקם את המתודות, המשתנים הפרטיים של המחלקות במיקום הטבעי בהיררכיה על מנת למנוע כפילויות מיותרות.
  - עימוד (indent) של הקוד.
- בתרגיל זה עליכם להקפיד:
  - שימוש במשתנים const, פונקציות const היכן שצריך ואם צריך. כולל בהעברה לפונקציה וכדומה.
  - שימוש נכון בהקצאות זכרון, ושחרור נכון של הזכרונות תוך התייחסות לכל מקרה אפשרי.
  - מימוש Copy Constructor וOperator= לכל מחלקה.
  - חלוקה נכונה של הפונקציות ל"private" ו"public".
- **ניתן להניח** שאם התוכנית מצפה למספר או למחרוזת הקלט יהיה בהתאם **אבל** אם מצפים למספר **לא ניתן להניח** קלט חיובי/שלילי, ויש להתייחס לכל המקרים.

הגדרת התוכנית

עליכם לתכנן תוכנית הממשת משחק TAKI.

מהלך התוכנית:

בשלב ראשון, נקבעים מספר השחקנים, ומספר הקלפים לכל שחקן על ידי קלט מהמשתמש.

בשלב שני, התוכנית מקבלת את השם של כל שחקן ומחלקת לו קלפים באופן אקראי. כמו כן התוכנית מגרילה קלף התחלתי. לאחר מכן מתחיל המשחק.

דוגמא:

```
How many players?
2
How many cards?
7
player number 1 name?
ELINA
player number 2 name?
ANNA
```

מהלך המשחק:

התוכנית תעבור על השחקנים לפי הסדר (אין חשיבות מי השחקן הראשון), בהגיע תורו של כל שחקן על התוכנית להציג שורה כזו:

```
current: 9
ELINA, your turn -
Your cards: (1)STP (2)3 (3)STP (4)8 (5)7 (6)8 (7)STP
```

כעת ישנן שלוש אפשרויות:

1. המשתמש בחר בחירה חוקית (אותו צבע, או אותה צורה) – במקרה כזה הקלף הנוכחי מוחלף לקלף הנבחר ומתקדם התור (בהתאם לקלף שנבחר, ראה להלן)

```
current: 3
ANNA, your turn -
Your cards: (1)6 (2)7 (3)2 (4)->&lt;- (5)+ (6)1 (7)3
```

2. המשתמש בחר בחירה לא חוקית של קלף – תופיע שורת שגיאה באופן הבא, והתוכנית תחכה לקלט אחר:

```
7
You can't put STP on 9
```

3. המשתמש בחר 0, או כל מספר מחוץ לטווח – השחקן הנוכחי מקבל קלף חדש מהקופה, והתור מתקדם בכל מקרה ב-1.

התקדמות התור באפשרות 1 נעשית בהתאם לחוקי המשחק:

- עבור קלף + - **אין התקדמות תור**
- עבור קלף מחליף כיוון – **שינוי כיוון המשחק, התקדמות ב -1**
- עבור קלף עצור – **התקדמות ב-2 (דילוג על השחק הבא)**
- עבור כל קלף אחר – **התקדמות ב-1**

המנצח הוא השחקן שסיים את הקלפים ראשון. לאחר מכן מודפס: `PlayerName wins!` בהתאם לשם השחקן והתכנית מסתיימת.

שימו לב – אין צורך להתייחס לאפשרויות אחרות (למשל מחליף צבע, TAKI, ו2+). מי שבכל זאת רוצה, מוזמן ☺

המימוש:

התוכנית תהיה מורכבת משלוש מחלקות:

1. מחלקת Card (ממומשת):

- אתחול על ידי צבע וסימן
- צבע וסימן מוגדרים על ידי enumים. כאשר צבע הוא אחד מתוך RGBY, וסימן הוא #N עבור מספר או אחד מ:
  - **PLUS** - סימן +
  - **CD**– שינוי כיוון סימן -><-
  - **STOP**- עצור
  - **TAKI**– משמש כקלף רגיל
- פונקציה Is_leggal מקבלת רפרנס לקלף אחר ובודקת האם ניתן לשים אותו על הקלף הנוכחי. מחזירה ערך בוליאני.
- בנוסף, המחלקה תומכת באופרטור הדפסה. כלומר בהינתן מופע של Card למשל c1 ניתן לכתוב:
  cout << c1;
  והקלף יודפס בצבע המתאים.
- כמו כן נתונה פונקציה generate_card המגרילה קלף ומחזירה אותו. **כל יצירת הקלפים בתוכנית תיעשה דרך הפונקציה הזו!**

2. מחלקת Player:

- מאותחלת על ידי שם השחקן , ומספר קלפים לשחקן.
- מכילה שדות:
  - שם
  - מספר קלפים
  - וקטור המכיל קלפים
- כמו כן תומכת בפונקציה ציבורית **play**. הפונקציה מקבלת רפרנס לקלף נוכחי. בתוך פונקציה זו מתבצע התור של השחקן. כשבסופו מוחזר ערך בוליאני: true אם השחקן שיחק, false אם הוא לקח קלף מהקופה.

3. מחלקת Game:

- מחלקה זו מנהלת את כל המשחק. שומרת את הקלף הנוכחי, רשימה דינמית של שחקנים, את התור הנוכחי, ואת כיוון התקדמות התור. בנוסף תומכת בפונקציה start שמתחילה את המשחק.
- עבור מחלקה Game ניתן להניח שיש מופע בודד בקוד, ולכן **אינכם נדרשים לממש copy constructor ו=operator**. עם זאת, **עליכם לוודא שלא יופעל** – גם לא בטעות – מימוש דיפולטיבי שלהם. (על ידי הצהרתו כפונקציה פרטית)

ניתן לראות פירוט של המחלקות בקבצי הקוד המצורפים.

לכל שאלה מוזמנים לפנות במייל.

בהצלחה!

הערות

- לצורך בדיקת הפתרון שלכם לפני ההגשה למערכת. תמחקו או תחליפו להערה את שורה 4 בקובץ Card.cpp:
  - `#define RANDOM → //#define RANDOM`

פעולה זו תגרום לכך שהקלפים יחולקו בסדר ידוע ולא אקראי. לאחר מכן תכניסו את הקלט הבא לפי הסדר משמאל לימין:

```
2 2 p1 p2 1 1 1 2 2 3 2 1
```

לאחר קלט זה המשחק אמור להסתיים! במקרה שהמשחק לא מסתיים תבדקו את עצמכם.
במקרה של הגשה למערכת כאשר המשחק לא מסתיים עבור קלט זה - לא יתקבל שום פלט והמערכת תחזיר תוצאה כזו: **0 points (Timeout)**

הפלט המצופה עבור קלט זה הוא:

```
How many players?
How many cards?
player number 1 name?
player number 2 name?
current: G-8
p1, your turn-
Your cards: (1)G-9 (2)R+-
current: G-9
p2, your turn-
Your cards: (1)G-STP (2)R-><--
current: G-STP
p2, your turn-
Your cards: (1)R-><--
You can't put R -><--on G-STP
current: G-STP
p1, your turn-
Your cards: (1)R+-
current: G-STP
p2, your turn-
Your cards: (1)R(2) -><--R-9
current: G-STP
p1, your turn-
Your cards: (1)R-+ (2)G+-
current: G+-
p1, your turn-
Your cards: (1)R+-
p1 wins!
```

- במקרה של הסרת קלף מהרשימה – יש להשאיר את שאר הקלפים לפי הסדר הקודם.
  במקרה של קבלת קלף חדש – יש להוסיפו בסוף הרשימה
- אין צורך להשתמש ב*char לשמירת מחרוזות. השתמשו בstring – פשוט הרבה יותר.
- אין להקצות מקום מיותר. כל הקצאה תהיה לפי הכמות הדרושה בדיוק.
- במקרה שיש צורך בהקצאה מחודשת. אין להשתמש בפונקציה realloc משפת C. יש לשחרר את הזכרון הישן ולהקצות מחדש, תוך שימוש בפקודות של שפת ++C.
- במקרה של קבלת מספר שלילי עבור מספר הקלפים או השחקנים יש להדפיס הודעת שגיאה כלשהי ולצאת מהמשחק – הפלט המדויק לא ייבדק, אלא רק ההתמודדות עם המקרה
- שימוש **בconst**. עליכם לוודא שכל אחת מהפונקציות שלכם שאינה משנה ערכים היא const. לא בהכרח שיש כאלו – זה תלוי מימוש.
- מקרי קצה שעליכם להתייחס:
  - כל המערכים שלכם מוקצים דינאמית – אין זיכרון שמוקצה סתם (אין "מספרי קסם" בתוכנית)
  - בהקצאות ושחרור להתייחס למקרים שהמערך ריק.
  - שימוש נכון באופרטור= וCopy Constructor

    - שימוש נכון בconst בכל התצורות שלו.
- מקרי קצה שלא ייבדקו:
    - מהלך משחק שונה מזה שנבדק בבדיקה האוטומטית
    - התנהגות משונה של המשתמש (למעט מה שצוין בפירוש)