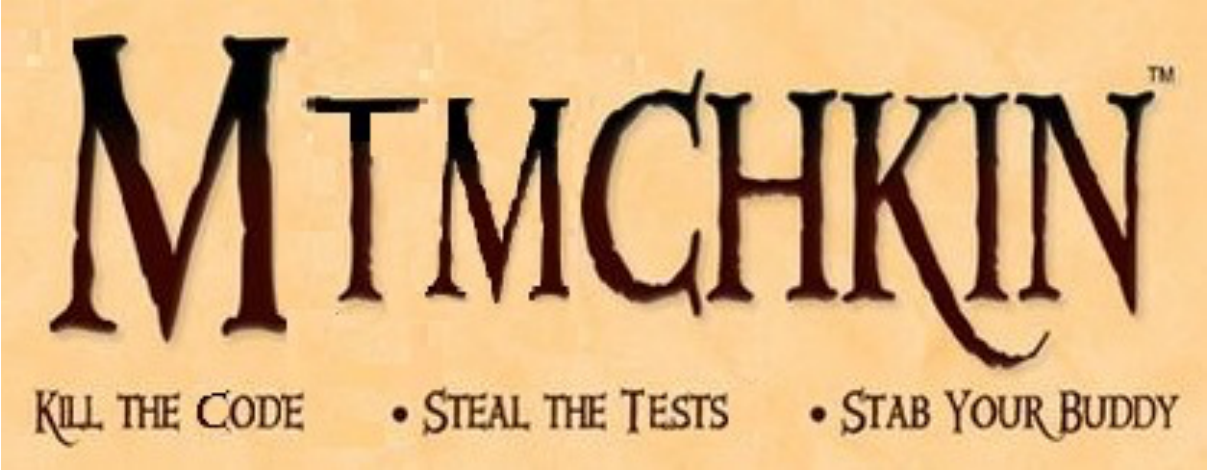
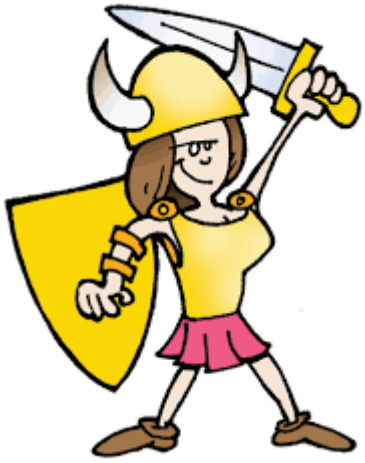


234124 - מבוא לתכנות מערכות
תרגיל בית מספר 2
סמסטר חורף 2022-2023

תאריך פרסום:27.11.2022
תאריך הגשה: 11.12.2022 בשעה 23:59
מתרגלים אחראים לתרגיל: אחלאם אבוגוש.

1 הערות כלליות

- **תרגיל זה מהווה 4% מהציון הסופי**
- התרגיל להגשה **בזוגות** בלבד.
- מענה לשאלות בנוגע לתרגיל יינתן אך ורק בפורום התרגיל בפיאצה או בסדנות. לפני פרסום שאלה בפורום אנא בדקו אם כבר נענתה – מומלץ להיעזר בכלי החיפוש שהוצגו במצגת האדמיניסטרציה בתרגול הראשון.
- שימו לב: לא תינתנה דחיות במועד הגשת התרגיל פרט למקרים חריגים. **תכננו את הזמן בהתאם**.
- ראו את התרגיל עד סופו לפני שאתן מתחילות לממש. חובה להתעדכן בעמוד ה-F.A.Q של התרגיל, הכתוב שם מחייב.
- העתקות קוד בין סטודנטים ובפרט גם העתקות מסמסטרים קודמים תטופלנה. עם זאת – מומלץ ומבורך להתייעץ עם חברים על ארכיטקטורת המימוש.
- קבצי התרגיל נמצאים בGitHub repository הבא: https://github.com/CS234124/ex2
- המסמך נכתב בלשון זכר מטעמי נוחות בלבד ומיועד לשני המינים.



2 תרגילי בית 2+3+4

בתרגילי בית 2+3+4 נממש משחק קלפים/תפקידים רב-משתתפים המבוסס (אך שונה) על משחק הקלפים הפופולרי Munchkin. במשחק זה תלחמו במפלצות, תדבגו באגים ותתקנו דליפות זיכרון על מנת לעלות בדרגות ולהגיש את תרגילי הבית בזמן.

3 MTMCHKIN

תרגיל בית 2 מהווה גרסה מצומצמת (ודטרמניסטית) של המשחק המיועדת לשחקן אחד.

3.1 *מהלך המשחק*

מטרת המשחק היא לנצח בקרבות ולהגיע לרמה 10. בכל תחילת סיבוב, השחקן שולף קלף מחפיסת הקלפים. לאחר שליפת הקלף, השחקן "נתקל" (encounter) בקלף. הקלף יכול להיות קלף קרב (Battle) בו צריך לנצח בקרב בכדי לעלות רמה, קלף שיפור (Heal/Buff) המעלה את הכוח/נקודות החיים תמורת מטבעות, או קלף אוצר (Treasure) אשר מעניק מטבעות. לאחר סיום ההיתקלות, נגמר הסיבוב.

למשחק יש 3 מצבים:

- MidGame – המשחק עדיין רץ. השחקן ממשיך לשלוף קלפים ולהיתקל בהם.
- Win – המשחק נגמר והשחקן ניצח, כלומר השחקן הגיע לרמה (level) 10.
- Loss – המשחק נגמר והשחקן הפסיד, כלומר נקודות החיים (HP – Health Points) ירדו ל-0.

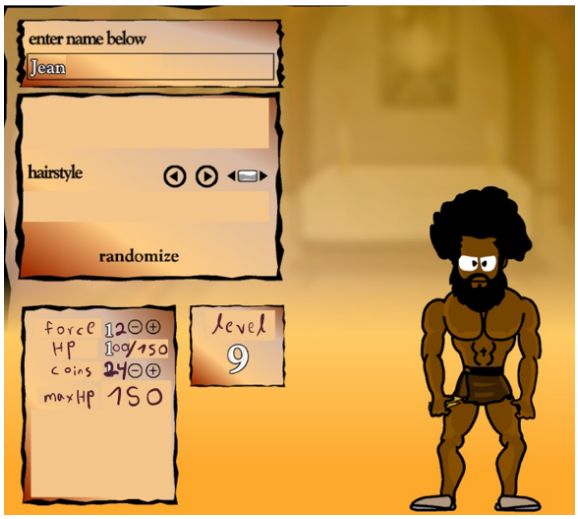
3.2 *סוגי קלפים*

קרב (Battle) - קלפים אלה יכולים להוביל ל-2 תוצאות:

- ניצחון בקרב אם כוח ההתקפה של השחקן גדול או שווה לכוחו (force) של קלף הקרב. במקרה זה השחקן יעלה רמה אחת, ויקבל מספר מטבעות (loot) המוגדר לקלף.
- הפסד בקרב אם כוח ההתקפה של השחקן קטן מכוח (force) קלף הקרב. במקרה זה נקודות החיים (HP – Health points) יפגעו במספר נקודות שהקלף מגדיר (עד למינימום של 0).

Heal/Buff – קלפי שיפורים. אם יש מספיק מטבעות על מנת לשלם את המחיר הספציפי (שמוגדר על ידי הקלף) עבור השיפור שהקלף מציע, אז השחקן משלם את המחיר ובתמורה מקבל את השיפור. ריפוי של נקודות החיים (HP – Health points) עבור קלף Heal, או שיפור כוח אם מדובר בקלף Buff.

Treasure - קלפי אוצר. מוסיפים לשחקן מספר מטבעות על פי המוגדר בקלף.



3.3 *שחקן/ית*

שחקן מאופיין ב:

- שם (name) - מורכב רק מאותיות באנגלית וללא רווחים.
- רמה (level) – מספר טבעי בטווח הערכים [1,10].
- כוח (force) – מספר טבעי אי שלילי.
- נקודות חיים מקסימליות (maxHP) – מספר טבעי חיובי.
- נקודות חיים (HP – Health Points) – מספר טבעי בטווח הערכים [0,maxHP].
- כמות מטבעות (coins) – מספר אי שלילי שלם.

שחקן התחלתי מתחיל ברמה (level) 1, עם 0 מטבעות (coins), ונקודות חיים (HP) מלאות, כלומר שוות לנקודות חיים המקסימליות (maxHP).

4 מימוש MTMCHKIN

4.1 *המחלקה Player*

תחילה נממש מחלקה שתייצג לנו שחקן במשחק. מאפייני השחקן ישמרו ויתוחזקו על ידי מחלקה זו. המחלקה תקרא Player.
עליכם להשלים את הממשק והמימוש של מחלקה זו בקבצים Player.h, ו-Player.cpp. ממשק המחלקה צריך להכיל את המתודות הבאות:

4.1.1 בנאי

מייצר שחקן ברמה 1, עם אפס מטבעות, כוח התחלתי שהתקבל כארגומנט לבנאי, ו-HP מלא ששווה ל-HP המקסימלי שהתקבל כארגומנט לבנאי. הכוח ההתחלתי הדיפולטי הוא 5, וה-HP המקסימלי הדיפולטי הוא 100.
במקרה והערכים המתקבלים בבנאי אינם תקינים, יש לאתחל את השדות לערכים הדיפולטיים.

```
Player player1("Efrat",150,2);  //Efrat has 150 max HP and 2 points of force.
Player player2("Gandalf",300);  //Gandalf has 300 max HP and 5 points of force.
Player player3("Daniel"); //Gandalf has 100 max HP and 5 points of force.
```

4.1.2 בנאי העתקה, הורס ואופרטור השמה

בנאי העתקה, הורס ואופרטור השמה מוגדרים באופן הסטנדרטי שראיתם בקורס.

4.1.3 הדפסת פרטי שחקן

הדפסת פרטי השחקן למסך. תוצאת הקוד הבא:

```
player2.printInfo();
```

יהיה בפורמט הבא:

```
Player Details:
Name: Gandalf
Level: 1
Force: 5
HP: 300
Coins: 0
------------------------
```

מסופקת לכם פונקציית עזר בשם printPlayerInfo המוגדרת בקובץ utilities.h. יש להיעזר בה על מנת להדפיס את פרטי השחקן בפורמט הרצוי.

4.1.4 עליית רמה

מעלה רמה אחת. אפשר להעלות את הרמה עד רמה 10, לאחר שמגיעים לרמה 10 המתודה אינה עושה דבר.

```
player2.levelUp();
```

4.1.5 תשאול רמה

מחזיר את רמת השחקן:

```
player2.getLevel();
```

לדוגמה, המתודה למעלה תחזיר 2. כי הרמה של player2 (גנדלף) היא 2 לאחר שבסעיף הקודם קראנו למתודה levelUp.

4.1.6 שיפורי כוח

מעלה את נקודות הכוח (force) בכמות המתקבלת בתור ארגומנט:

```
player2.buff(1);
```

הקוד למעלה יעלה את הכוח של player2 (גנדלף) ל6.

4.1.7 ריפוי HP

מעלה את נקודות החיים (HP) של השחקן בכמות המתקבלת בתור ארגומנט, עד למקסימום של maxHP:

```
player3.heal(10);
```

הקוד למעלה **לא** יעלה את נקודות החיים של player3 (דניאל) ב-10 מכיוון שנקודות החיים שלו שוות כבר לmaxHP. אם הפרמטר לפונקציה קטן מאפס הפונקציה לא תבצע דבר.

4.1.8 פגיעה בHP

מוריד את נקודות החיים (HP) של השחקן בכמות המתקבלת התור ארגומנט, עד ל-0:

```
player3.damage(10);
```

הקוד למעלה יוריד את נקודות החיים של player3 (דניאל) ב10. אם הפרמטר לפונקציה קטן מאפס הפונקציה לא תבצע דבר. אם הפרמטר לפונקציה גדול מה-HP הנוכחי של השחקן, הפונקציה תוריד את ה-HP של השחקן לאפס.

4.1.9 בדיקת HP

בודק אם כמות נקודות החיים (HP) הגיעה לאפס:

```
if (player2.isKnockedOut()){
     std::cout << "Player is knocked out";
}
```

4.1.10 הוספת מטבעות

הוספת מטבעות לשק המטבעות. בקוד הבא מתווספים 10 מטבעות לגנדלף:

```
player2.addCoins(10);
```

4.1.11 תשלום

תשלום מטבעות יתבצע על ידי המתודה pay, אשר תוריד מטבעות משק המטבעות.
מוחזר true כאשר התשלום הצליח, אחרת מוחזר false והתשלום לא מתבצע כלל (אין שינוי בכמות המטבעות).
הקוד הבא ידפיס "Not enough Coins", כי ל-player1 יש 0 מטבעות:

```
if (!player1.pay(10)){
    std::cout << "Not enough coins";
}
```

4.1.12 כוח התקפה

מחזיר את כוח ההתקפה. כוח ההתקפה מוגדר להיות הכוח (force + הרמה (level)). לדוגמה הקריאה הבאה עבור player2 (גנדלף) תחזיר 8 מכיוון שיש לו 6 נקודות כוח, והוא ברמה 2.

```
player2.getAttackStrength()
```

4.2 המחלקה Card

המחלקה Card מייצגת קלף במשחק. ממשק המחלקה נתון בקובץ המסופק Card.h (אין לשנותו) ומתואר בקובץ ובתתי הסעיפים הבאים. עליכם לממש את הממשק בקובץ Card.cpp.

למחלקה יש 2 מאפיינים המתקבלים מהמשתמש בעת האתחול:

- CardType - סוג הקלף (מתקבל בעזרת enum class בקובץ Card.h).
- CardStats – נתוני הקלף, פרמטרים מספריים של הקלף המתקבלים כמבנה מסוג CardStats שמוגדר ב-utilities.h.

נתוני קלף:

שימו לב כי רק חלק מן הנתונים המספריים שמוגדרים ב-CardStats רלוונטיים לכל סוג של קלף! סוגי הקלף להם יש שימוש בשדה נמצאים בתוך [].

- loot – מספר המטבעות שיש להוסיף לשחקן בעת ניצחון בקרב, או בעת היתקלות בקלף Treasure. [Battle, Treasure]
- force – כוח הקלף. [Battle]
- hpLossOnDefeat – מספר נקודות החיים (HP – Health points) שיורדות בעת תבוסה בקרב. [Battle]
- cost – מחיר קלף שיפורים. [Heal, Buff]
- heal – כמות נקודות החיים שקלף מסוג heal מעלה. [Heal]
- buff – כמות נקודות כוח שקלף מסוג buff מעלה. [Buff]

4.2.1 בנאי

מייצר קלף מסוג CardType עם ה-CardStats המתקבלים כארגומנטים:

```
CardStats stats(3, 40, 10, 30, 1, 20);
Card card(CardType::Battle,stats);
```

4.2.2 התקלות (encounter)

המתודה מקבלת שחקן הנתקל בקלף, מבצעת את פעולת הקלף כמפורט ב- **3.2**, ומשנה את השחקן בהתאם. בהיתקלות בקלף מסוג Battle, המתודה תדפיס את תוצאות הקרב על ידי קריאה לפונקציה printBattleResult אשר מוגדרת ב-utilities.h

```
card.applyEncounter(player);
```

4.2.3 הדפסת פרטי קלף

המתודה PrintInfo תדפיס את פרטי הקלף בעזרת הפונקציות המתאימות בutilities.h: printBattleCardInfo, printBuffCardInfo, printHealCardInfo, printTreasureCardInfo.

```
card.printInfo();
```

בדוגמה לעיל יודפס:

```
Card drawn:
Type: Battle
Force: 3
Profit (on win): 20
Damage taken (on loss): 40
------------------------
```

4.3 המחלקה *Mtmchkin*

המחלקה Mtmchkin מנהלת את המשחק. המחלקה מחזיקה את השחקן, את חפיסת הקלפים ואת סטטוס המשחק. הממשק שלה נתון בקובץ החלקי המסופק Mtmchkin.h (יש להשלים את השדות של Mtmchkin ואת מימוש הממשק. בנוסף, בהתאם לצורך אתם יכולים גם להוסיף בנאים, הורסים וכו').
בתחילת המשחק, מאתחלים את המשחק עם חפיסת קלפים שנתונה במערך. בכל סיבוב ניגשים לקלף הבא בחפיסת הקלפים, מאינדקס אפס בסיבוב הראשון עד סוף המערך, ובצורה מעגלית חוזרים להוציא קלפים מתחילת החפיסה כאשר מגיעים לסופה.

4.3.1 בנאי

הבנאי מקבל 3 ארגומנטים – שם השחקן, מערך קלפים, גודל מערך הקלפים.

```
Card cards[4];
CardStats stats(3, 40, 10, 30, 1, 20);
cards[0] = Card(CardType::Treasure,stats);
cards[1] = Card(CardType::Buff,stats);
cards[2] = Card(CardType::Battle,stats);
cards[3] = Card(CardType::Heal,stats);
Mtmchkin game("Daniel", cards, 4);
```

המשחק מאותחל בבנאי כך שמצב המשחק הוא MidGame, הקלף הבא שיקרא הוא הקלף הראשון במערך הקלפים, והשחקן עם הערכים הדיפולטים של כוח ו-maxHP.

4.3.2 לשחק את הקלף הבא (סיבוב במשחק)

בכל סיבוב יש לשלוף את הקלף הבא מחפיסת הקלפים, לבצע את ההיתקלות בו ולהדפיס את המידע התואם. אם הגענו לקלף האחרון, חוזרים לתחילת החפיסה ושולפים שוב את הקלף הראשון.

```
game.playNextCard();
```

שלבי הסיבוב:

1. שליפת הקלף הבא מחפיסת הקלפים.
2. הדפסת פרטי הקלף.
3. ביצוע ההיתקלות בקלף.
4. הדפסת פרטי השחקן העדכניים.

מצורף פלט לדוגמה עבור היתקלות בקלף Battle אשר נגמר בניצחון:

```
Card drawn:
Type: Battle
Force: 3
Profit (on win): 20
Damage taken (on loss): 40
------------------------
The player defeated the monster and won the loot! Hooray!
------------------------
Player Details:
Name: Daniel
Level: 2
Force: 6
HP: 100
Coins: 30
------------------------
```

4.3.3 תשאול סיום משחק

מחזיר true אם המשחק הסתיים בניצחון או הפסד, אחרת false.

```
while(!game.isOver()){
    game.playNextCard();
}
```

4.3.4 החזרת מצב המשחק

getGameStatus תחזיר את מצב המשחק.

```
if (game.getGameStatus() == GameStatus::Win){
   std::cout << "The player defeated all monsters and achieved eternal glory!";
}
```

4.4 *דגשים ודרישות מימוש*

- קראו את התיעוד ב-Card.h ו-Mtmchkin.h!
- מסופקים לכם טסטים בסיסיים עבור המחלקות השונות בקובץ test.cpp.
- המימוש חייב לציית לכללי כתיבת הקוד המופיעים תחת Course Material -> Code Conventions.
  אי עמידה בכללים אלו תגרור הורדת נקודות.
- בניגוד לשמועות, הקונבנציה עבור שדות מחלקה היא להתחיל את שמן עם m_<name>.
- על המימוש שלכם להתבצע ללא שגיאות זיכרון (גישות לא חוקיות וכדומה) וללא דליפות זיכרון.
- המערכת צריכה לעבוד על שרת CSL3 או CSL33.
- אתם לא צריכים להשתמש בחריגות בפתרון התרגיל (תרגול 6).
- **אסור להשתמש ב-STD פרט ל-std::string!!**

4.5 *הידור קישור ובדיקה*

התרגיל ייבדק על שרת csl3 ועליו לעבור הידור בעזרת הפקודה הבאה:

```
> g++ --std=c++11 -o mtmchkin_test -Wall -pedantic-errors -Werror -DNDEBUG *.cpp
```

לנוחיותכם, מסופקת לכם תוכנית "בדיקה עצמית" בשם finalCheck, התוכנית בודקת שקובץ ההגשה, קובץ ה-zip בנוי נכון ומריצה את הטסטים סופקו כפי שירוצו על ידי הבודק האוטומטי. הפעלת התוכנית ע"י הפקודה:

```
~mtm/public/2223a/ex2/finalCheck <submission>.zip
```

הקפידו להריץ את הבדיקה על קובץ ההגשה. אם אתה משנים אותו, הקפידו להריץ את הבדיקה שוב.

4.6 *ולגרינד ודליפות זיכרון*

המערכת חייבת לשחרר את כל הזיכרון שעמד לרשותה בעת ריצתה. על כן עליכם להשתמש ב-valgrind שמתחקה אחר ריצת התכנית שלכם, ובודק האם ישנם משאבים שלא שוחררו. הדרך לבדוק האם יש לכם דליפות בתכנית היא באמצעות שתי הפעולות הבאות (שימו לב שחייב להיות main, כי מדובר בהרצה ספציפית):

1. קימפול של השורה לעיל עם הדגל g-.
2. הרצת השורה הבאה:

   valgrind --leak-check=full ./mtmchkin_test

   כאשר mtmchkin_test הוא שם קובץ ההרצה.

הפלט ש-valgrind מפיק אמור לתת לכם, במידה ויש לכם דליפות, את שרשרת הקריאות שהתבצעו וגרמו לדליפה. אתם אמורים באמצעות דיבוג להבין היכן היה צריך לשחרר את אותו משאב שהוקצה ולתקן את התכנית. בנוסף, valgrind מראה דברים נוספים כמו פנייה לא חוקית לזיכרון (שלא בוצע בעקבותיה segmentation fault) – גם שגיאות אלו עליכם להבין מהיכן הן מגיעות ולתקן.

4.7 *בדיקת התרגיל*

בדיקה יבשה כוללת מעבר על הקוד ובודקת את איכות הקוד והתכן (שכפולי קוד, קוד מבולגן, קוד לא ברור, שימוש בטכניקות תכנות "רעות").
בדיקה רטובה כוללת את הידור התכנית המוגשת והרצתה במגוון בדיקות אוטומטיות. על מנת להצליח בבדיקה שכזו, על התוכנית לעבור הידור, לסיים את ריצתה, ולתת את התוצאות הצפויות ללא דליפות זיכרון.

4 הגשה

את ההגשה יש לבצע דרך אתר הקורס, תחת
Assignments -> HW2 -> Electronic Submit.

הקפידו על הדברים הבאים:

- יש להגיש את קבצי הקוד מכווצים לקובץ zip (לא פורמט אחר).
- **אין להגיש אף קובץ מלבד קבצי h וקבצי cpp אשר כתבתם או השלמתם בעצמכם.**
- הקבצים אשר מסופקים לכם יצורפו על ידנו במהלך הבדיקה, וניתן להניח כי הם יימצאו בתיקייה הראשית.
- ניתן להגיש את התרגיל מספר פעמים, רק ההגשה האחרונה נחשבת.
- על מנת לבטח את עצמכם נגד תקלות בהגשה האוטומטית שימרו את קוד האישור עבור ההגשה. עדיף לשלוח גם לשותף. כמו כן שימרו עותק של התרגיל על חשבון ה-Google Drive/Github/CSL3 שלכם לפני ההגשה האלקטרונית ואל תשנו אותו לאחריה (שינוי הקובץ יגרור שינוי חתימת העדכון האחרון).
  - כל אמצעי אחר לא יחשב הוכחה לקיום הקוד לפני ההגשה.

```
submission.zip
├── Player.h
├── Player.cpp
├── Card.cpp
├── Mtmchkin.h
├── Mtmchkin.cpp
└── ...possibly other files
```

בהצלחה !