מבוא לתכנות מערכות – 234124

תרגיל בית 4

סמסטר חורף 2025

תאריך פרסום: 8.1.25
תאריך אחרון להגשה: 29.1.25 עד השעה 23:55
מתרגל אחראי לתרגיל: ינון גולדשטיין

1 הערות כלליות

- תרגיל זה מהווה 8% מהציון הסופי בקורס.
- התרגיל להגשה בזוגות בלבד.
- כל ההודעות והעדכונים הנוגעים לתרגיל זה יפורסמו באתר הקורס ב- Webcourse.
- מענה לשאלות בנוגע לתרגיל יינתן אך ורק בפורום התרגיל בפיאצה (קישור באתר הקורס) או בסדנאות. אין לשלוח דוא"ל לגבי התרגיל, **מיילים לגבי התרגיל לא יענו**.
- שימו לב – לא תינתנה דחיות למועד הגשת התרגיל.
- העתקות קוד בין סטודנטים תטופלנה בחומרה. עם זאת – מומלץ ומבורך להתייעץ עם חברים על ארכיטקטורת המימוש.
- על כל הקוד שאתם כותבים לעמוד בקונבנציות המפורטות בקובץ "קונבנציות לכתיבת קוד" הנמצא באתר הקורס. אי עמידה בקונבנציות עלולה לגרור הורדת ניקוד.
- קבצי התרגיל מסופקים לכם ב-GitHub.
- החומר הנדרש כדי לפתור את התרגיל הינו החומר הנלמד עד הרצאה 10 ועד ותרגול 8 (כולל). לצורך פתרון התרגיל מותר להשתמש בכל תכונה שנלמדה בקורס בשפת C++.
- תיקונים למסמך התרגיל, יסומנו בצהוב.

רשימת התיקונים בקובץ:
תיקון 1 – מה קורה כשדמות שנלחמת בטווח קצר כמו לוחם מנצחת/מפסידה?
תיקון 2 – מה קורה לדמות שיורדות לה נקודות כוח כשנקודות הכוח שלה הן כבר 0?
תיקון 3 – האם דמויות עם 0 חיים אמורות להופיע ברשימת הניקוד?
תיקון 4 – איך מדפיסים נכון תוכן של להקה עם תתי-להקות?
תיקון 5 – פורמטים אפשריים של תוכן קובץ אירועים

בנוסף, הפוסט העליון הנעוץ בפיאצה מכיל הערות ותיקונים שעונים על שאלות נפוצות שהופיעו בפיאצה.
זהו **פוסט מחייב שחובתכם לקרוא** ויורדו נקודות למי שלא יעמוד בתוכן שפורסם בו.

2 הקדמה

בתרגיל בית זה תתכננו ותממשו מערכת מורכבת. מעבר לכך שהמימוש שלכם צריך לעבוד ולעבור בדיקות אוטומטיות, אנחנו שמים דגש על האופן בו אתם בוחרים את התכן (Design) של הרכיבים השונים במערכת, היכן אתם בוחרים להשתמש בתבניות תכן (Design Patterns) והאם אתם משתמשים במנגנוני השפה באופן המתאים לתכן שלכם.

המלצה לסדר עבודה על התרגיל:

1. קראו את היטב את הוראות התרגיל ואת סיפור הרקע על המערכת, נסו להבין את הקשרים השונים הנדרשים בין חלקי המערכת.
2. התחילו מלפתור את החלק היבש של התרגיל (מומלץ יחד) – בחלק היבש נבקש מכם לתאר את התכן שלכם ונשאל שאלות מנחות על המערכת. תכנון טוב של המערכת לפני תחילת המימוש יקל עליכם פלאים בזמן המימוש ויחסוך לכם זמן יקר!
3. כתבו את המימוש לחלקים השונים של המערכת. יכול להיות שבזמן המימוש תבינו שצריך לחזור אחורה ולעשות שינויים בתכן, זה בסדר והגיוני.

טיפים נוספים:

- תכננו את המערכת שלכם בצורת Top-Down Design – התחילו מלהבין את הדרישות והמשימות של המערכת הכוללת ופצלו אותן לדרישות קטנות יותר, עד שאתם מגיעים למשימות שקל לממש.
- כשמתחילים לממש - התחילו מלהגדיר את הממשקים השונים בין החלקים של המערכת, מה המתודות שיש לכל מחלקה ואיזו מחלקה אחראית על איזו פעולה. תחשבו מה העבודה שכל חלק אמור לעשות!
- חלקו את העבודה בין השותפים - חשוב ששני השותפים יכירו את התכן ואת המימוש, אך יחד עם זאת חשוב שתהיו יעילים בזמן.
- אל תחכו לרגע האחרון כדי להתחיל לעבוד על התרגיל! התרגיל הזה ארוך ולוקח זמן, נצלו את הזמן שיש לכם כמה שרק אפשר.
- נסו לבדוק את התרגיל שלכם מספר פעמים לאורך המימוש ולאחר כל חלק שאתם מסיימים לממש. כאשר אנחנו מבצעים בדיקות לאורך הפיתוח, אנחנו בודקים חלקים קטנים בכל פעם מה שמקל עלינו לתקן באגים ומעלה את רמת הבטחון שלנו בנכונות של המימוש. אם תגיעו לסוף המימוש ורק אז תתחילו לבדוק את התרגיל, יכול להיות שתמצאו באגים שכבר קשה לתקן בשלב מאוחר בפיתוח.

3 MatamStory

לאחר שפלטפורמת המטבעות הוירטואלים, סטודיו הסרטים ומערכת ניהול המשימות של מתרגלי הקורס לא צלחו, החליטו המתרגלים לנסות להיכנס לתחום פיתוח משחקי המחשב.

אולם, מכיוון שיצירתיות היא לא הצד החזק שלהם, החליטו המתרגלים לשאוב השראה ממשחק המחשב המוכר והאהוב MapleStory (שבמקרה מפותח גם הוא ב-++C). בתרגיל זה תעזרו למתרגלים לממש משחק בשם MatamStory המבוסס קלות על MapleStory (אין צורך להכיר את המשחק עבור פתרון התרגיל).

3.1 תיאור וחוקי המשחק

המשחק הינו מרובה שחקנים, יכולים להשתתף בו **שניים עד שישה** משתתפים, ומשוחק בתורות. בתחילת המשחק, כל שחקן בוחר את סוג העבודה (Job) של הדמות אותה הוא רוצה לשחק, ואת סוג האופי (Character) של הדמות שלו. העבודה והאופי יקבעו לאורך המשחק כיצד הדמות תתמודד עם אירועים שונים.

המשחק מחולק לסיבובים (Rounds). בכל סיבוב, כל אחד מהשחקנים מבצע תור (Turn) אחד, לפי סדר קבוע. בכל תור, השחקן שזהו התור שלו חווה אירוע (Event). האירוע יכול להיות היתקלות עם מפלצת (Encounter) או אירוע מיוחד (Special Event) כלשהו.

כאשר דמות נתקלת במפלצת (Encounter), היא צריכה לעמוד מולה בקרב. יכולות הקרב (CombatPower) של הדמות נקבעות על ידי מספר פרמטרים ומושפעות מהעבודה של הדמות. במידה ויכולת הקרב של הדמות טובה יותר מזו של המפלצת היא מנצחת בהיתקלות, עולה רמה ומרוויחה מטבעות זהב כפרס (Loot), אחרת היא מפסידה בקרב ומאבדת נקודות חיים (Health Points). אם נקודות החיים של הדמות הגיעו ל-0, הדמות מתעלפת (Knocked Out) יוצאת מהמשחק ולא משתתפת בו החל מהסיבוב הבא.

אירועים מיוחדים מייצגים אירועים שונים שהדמות יכולה להיקלע אליהם. לכל אחד מהאירועים האלו יש השפעה שונה על הדמות, ההשפעה והתגובה של הדמות מושפעות גם מהעבודה ומההתנהגות שלה.

לאחר כל סיבוב, אם אחד השחקנים הגיע לרמה 10 המשחק נגמר ומוכרז מנצח. במידה וכל השחקנים התעלפו ויצאו מהמשחק, המשחק מסתיים ללא מנצחים.

3.2 דמויות

דמות במשחק מאופיינת על ידי המאפיינים הבאים:

- שם (Name) – מורכב מאותיות באנגלית (קטנות וגדולות) בלבד, ללא רווחים, באורך עד 15 תווים
- רמה (Level) – מספר טבעי בטווח [1, 10].
- נקודות כוח (Force) – מספר שלם אי-שלילי.
- נקודות חיים נוכחיות (Current HP) – מספר שלם אי-שלילי, לא גבוה מכמות נקודות החיים המקסימליות.
- נקודות חיים מקסימליות (Max HP) – מספר שלם חיובי.
- כמות מטבעות (Coins) – מספר שלם אי שלילי.

בנוסף, כפי שתואר קודם לכן, כל דמות מאופיינת גם בעבודה (Job) ובאופי (Character) הנבחרים על ידי השחקן בתחילת המשחק.
כל עוד לא נאמר אחרת, כל דמות מתחילה את המשחק ברמה 1, עם 10 מטבעות, 5 נקודות כוח, 100 נקודות חיים מקסימליות וכמות נקודות חיים נוכחית השווה למקסימלית האפשרית לדמות.

3.2.1 עבודות

העבודה של הדמות יכולה להיות אחת מהשלוש הבאות:

- לוחם (Warrior) – לוחמים הם דמויות חזקות הנלחמות עם האויבים שלהן מטווח קרוב. יכולת הקרב של לוחמים במהלך היתקלות מחושבת לפי הנוסחה:
$CombatPower = Force * 2 + Level$
בנוסף, כמות נקודות החיים המקסימלית של לוחמים היא 150.

- קשת (Archer) – קשתים הם דמויות זריזות המסוגלות לפגוע באויבים שלהן ממרחק גדול.
בנוסף, קשתים מתחילים את המשחק עם 20 מטבעות (במקום 10).

- קוסם (Magician) - לדמויות אלו יכולות קסומות שעוזרות להן לקבל בונוסים שונים בעת אירועים קסומים (למשל ליקוי חמה). כמו הקשתים, גם קוסמים תוקפים את האויבים שלהם ממרחק.

3.2.2 סוגי אופי

כל שחקן יוכל לבחור לדמות שלו את אחד מסוגי האופי הבאים:

- אחראי (Responsible) - כאשר מוצגת בפני דמות אחראית בחירה כלשהי, היא תמיד תיקח את הבחירה הבטוחה יותר.
- לוקח סיכונים (RiskTaking) - כאשר מוצגת בפני דמות לוקחת סיכונים בחירה, היא תעדיף לקחת את האפשרות שמביאה לה רווח אפשרי גבוהה, גם אם כרוך בכך סיכון.

3.3 אירועים

בכל תור, הדמות שזהו תורה חובה אירוע כלשהו. אירוע יכול להיות היתקלות עם מפלצת או אירוע מיוחד.

3.3.1 מפלצות

באירועים מסוג היתקלות (Encounter), הדמות נתקלת במפלצת כלשהי (או קבוצה של מפלצות).
כל מפלצת מאופיינת על ידי:

- יכולת קרב (CombatPower) – מספר שלם אי-שלילי המעיד כמה המפלצת קשה להתמודדות.
- שלל (Loot) – כמות המטבעות שהדמות מקבלת במידה והיא מנצחת בהיתקלות.
- נזק (Damage) – כמות נקודות החיים שהדמות מאבדת במידה והיא מפסידה בהתקלות.

אלא אם נאמר אחרת, יכולת הקרב של דמות מחושבת על ידי הנוסחה:

$$CombatPower = Force + Level$$

במידה ויכולת הקרב של הדמות גבוהה יותר משל המפלצת, הדמות מנצחת, עולה רמה אחת וזוכה בכמות מטבעות השווה לשלל של המפלצת. אחרת, הדמות מפסידה ומאבדת נקודות חיים ככמות הנזק שהמפלצת עושה. דמויות הנלחמות מטווח קרוב (למשל לוחמים) מאבדות 10 נקודות חיים גם במקרה של נצחון.
במקרה של הפסד של דמות שנלחמת מטווח קצר, היא תאבד כמות חיים בכמות השווה לנזק שהמפלצת עושה (ללא תוספת של 10 נזק)

בתור התחלה, המשחק שלנו יכיל ארבעה סוגי מפלצות:

- חילזון (Snail) – עם יכולת קרב 5, שלל 2 ונזק 10.
- רפש (Slime) – עם יכולת קרב 12, שלל 5 ונזק 25.
- באלור (Balrog) – עם יכולת קרב 15, שלל 100 ונזק 9001.
  לאחר מפגש, יכולת הקרב של הבלרוג עולה ב-2, ללא קשר אם ניצח או הפסיד בקרב.
- להקה (Pack) – קבוצה של לפחות שתי מפלצות. יכולת הקרב, השלל והנזק שווים לסכום יכולות הקרב, השלל והנזק של חברי הלהקה.
  חברי הלהקה יכולים להיות מפלצות מסוגים שונים, כמו כן כל להקה יכולה להכיל תתי-להקות.

3.3.2 אירועים מיוחדים

במהלך התור, הדמות יכולה לחוות אירוע מיוחד, במקום להיתקל במפלצת. האירוע המיוחד משפיע רק על הדמות שזהו התור שלה.
בתור התחלה, במשחק שלנו יהיו שני אירועים מיוחדים אפשריים:

- ליקוי חמה (SolarEclipse) - אירוע קסום ונדיר המשפיע על כל דמות באופן שונה בהתאם לעבודה שלה. כאשר דמויות קסומות (למשל קוסמים) חווים ליקוי חמה היכולות הקסומות שלהם מתחזקות ומתווספת להם נקודת כוח אחת לשארית המשחק. דמויות אחרות שלא יודעות לנצל את האירוע הנדיר מתבלבלות מהחושך ומאבדות נקודת כוח אחת. דמות שמאבדת כך (או בכל צורה אחרת) נקודות כוח שאמורות להביא את סך נקודות הכוח שלה לקטן מאפס – מביא את נקודות הכוח שלה לאפס.
- סוחר שיקויים (PotionsMerchant) – הדמות נתקלת בסוחר שיקויים שמציע לה לקנות ממנו שיקויים המעלים את כמות נקודות החיים. העלות של כל שיקוי היא 5 מטבעות זהב וכל שיקוי מעלה לדמות 10 נקודות חיים. הדמות מחליטה כמה שיקויים לקנות בהתאם לאופי שלה:
  - דמות אחראית – תקנה כמה שיקויים שהיא יכולה עד שנגמרים לה מטבעות הזהב או שנקודות החיים שלה התמלאו לגמרי.
  - דמות לוקחת סיכונים – תקנה עד שיקוי אחד בלבד, ואך ורק אם כמות החיים הנוכחית שלה מתחת ל-50.

3.4 מהלך המשחק

בחלק זה נפרט עוד על מהלך המשחק.

3.4.1 קליטת האירועים והשחקנים

לפני תחילת המשחק, תחילה נקלט כל המידע הדרוש כדי לבנות את רשימת האירועים ואז נקלט כל המידע הדרוש לבנות את רשימת השחקנים.
המידע על האירועים ועל השחקנים נקלט מקבצים, עוד על הפורמט של הקבצים בהמשך.
לאחר סיום הקליטה, מתחילים את הסיבוב הראשון של המשחק, וממשיכים לשחק סיבוב אחר סיבוב עד שהמשחק נגמר.

3.4.2 מהלך סיבוב

בכל סיבוב, הדמויות ישחקו לפי הסדר בו הם נקלטו, כל אחת משחקת תור בודד. במידה ואחת מהדמויות התעלפה (נקודות החיים שלה ירדו ל-0) במהלך הסיבוב, היא יוצאת מהסבב ומפסיקה לשחק במשחק.
בסוף כל סיבוב תתעדכן טבלת הדירוג של השחקנים השונים ותוצג על המסך. דמויות שנקודות החיים שלהן ירדו ל-0 עדיין אמורות להופיע בטבלת הדירוג לפי הדירוג המחושב.

3.4.3 מהלך תור

בכל תור, דמות שזהו התור שלה חווה את האירוע הבא ברשימה (היתקלות או אירוע מיוחד). לאחר שהדמות שיחקה בהתאם לכללי האירוע, התור עובר לדמות הבאה. במידה והדמויות נתקלו בכל האירועים שנקלטו בתחילת המשחק, האירועים יחזרו להופיע לפי סדר קליטתם החל מהאירוע הראשון.

3.4.4 סיום המשחק

בסוף סיבוב, אם לפחות אחת מהדמויות הגיעה לרמה 10 או שכל הדמויות במצב מעולף, המשחק מגיע לסיומו.

במידה וקיימת דמות ברמה 10, הדמות הנמצאת בראש טבלת הדירוג מוכרזת בתור המנצחת במשחק. אחרת, המשחק מסתיים ללא מנצחים.

3.5 דירוג השחקנים – Leaderboard

כפי שהזכרנו קודם, בסוף כל סיבוב מוצגת טבלת הדירוג של המשתתפים. הטבלה מציגה את השחקנים בסדר ממויין לפי הכללים הבאים:

1. רמת הדמות (Level) – הדמות ברמה הכי גבוהה ראשונה
2. כמות מטבעות – במידה ויש שתי דמויות או יותר באותה רמה, הדמות עם כמות המטבעות הגדולה יותר תדורג גבוה יותר
3. שם הדמות – במידה ויש שתי דמויות או יותר באותה רמה ועם אותה כמות מטבעות, הן ידורגו לפי השם שלהן בסדר לקסיקוגרפי עולה (הדמות עם השם הקודם לפי סדר לקסיקוגרפי מופיעה ראשונה)

4 חלק יבש

בחלק זה תידרשו לתכנן את המערכת שלכם ולענות על שאלות על התכן. על התשובות שלכם בחלק זה להיות תואמות לפתרון החלק הרטוב.

4.1 תכן ושאלות יבשות

את התשובות לחלק זה יש לכתוב בקובץ dry.pdf.
לפני המענה על השאלות (ולפני תחילת המימוש), תכננו את המערכת שלכם. על המערכת להיות פתוחה להרחבה ככל הניתן מבלי שידרשו שינויים בקוד קיים ככל הניתן (ראו Open-Closed Principle). תכננו את המערכת כך שיהיה קל להוסיף עבודות חדשות, סוגי אופי ומפלצות חדשות בעתיד.

1. שרטטו דיאגרמת מחלקות ב-UML הכוללת את כל המחלקות הנדרשות למימוש הפתרון שלכם, כפי שלמדנו בהרצאות. אין צורך לכלול בדיאגרמה מחלקות המייצגות containers (למשל vector). יש לכתוב את שמות המחלקות בלבד ללא השדות והמתודות של המחלקה.

2. האם השתמשתם באיזשהן תבניות תכן (Design Patterns)? אם כן פרטו באילו השתמשתם והיכן בתכן השתמשם בהן.

3. נניח ונרצה להוסיף עבודה חדשה למשחק בשם גנב (Rogue). במידה ודמות מסוג גנב נתקלת במפלצת עם יכולת קרב גבוהה משל הדמות פי 2 או יותר, הדמות מתחמקת מהקרב ויוצאת ללא פגע.
ביצד הייתם מממשים את הדרישה החדשה? אילו דברים צריך להוסיף למערכת? אילו מקומות קיימים במערכת צריך לשנות?

4. נניח ונרצה להוסיף אירוע מיוחד חדש למערכת בשם השראה שמיימית (DivineInspiration). דמות החווה אירוע מסוג זה מחליפה את העבודה הנוכחית שלה בעבודה רנדומלית חדשה (ושמירה על בונוסים שצברה כגון ליקוי חמה לקוסם). האם ניתן לממש את הדרישה החדשה במערכת שלכם? אם כן – הסבירו אילו דברים צריך להוסיף או לשנות כדי לתמוך בדרישה. אם לא – הסבירו מדוע.

Git 4.2

בתרגיל זה, כמו בכל שאר התרגילים בקורס, עליכם להשתמש ב-Git במהלך העבודה על התרגיל. הקפידו על עבודה נכונה עם Commits ו-Branches כפי שנלמדה בהרצאות ובתרגולים.

בסיום העבודה על התרגיל, עליכם להדפיס את ה-log של ה-Repo לקובץ log.txt באמצעות הפקודה הבא:

```
git log --graph --all --stat > log.txt
```

צרפו את הקובץ log.txt ל-Repo (על ידי הוספת Commit נוסף) על מנת להגיש אותו.

5 חלק רטוב

בחלק זה תממשו את המשחק שתואר בתחילת המסמך (סעיף 3).

5.1 ממשקים נתונים

בקבצים המסופקים לכם קיימות מספר מחלקות להן הגדרנו עבורכם ממשק חלקי. מותר ואף צריך להוסיף לממשקים אלו הגדרות נוספות בהתאם לצרכי התכן שהגדרתם למערכת. בחלק זה נפרט את התכונות המינימליות בממשקים הנתונים, אותן אסור לכם לשנות.

5.1.1 MatamStory

מחלקה זו מנהלת את מערכת המשחק. ממשק המחלקה חייב להכיל לפחות את שני הדברים הבאים:

- בנאי – למחלקה בנאי אחד ויחיד המקבל שני ערוצי קלט (istream). ערוצים אלו מכילים את המידע על האירועים ועל השחקנים.
- המתודה play – כאשר נקרא למתודה זו, המשחק יתחיל וירוץ עד סופו.

5.1.2 Player

מחלקה זו מייצגת דמות\שחקן במשחק. הממשק המחייב למחלקה זו כולל את המתודות הבאות:

- getName – מחזירה את שם הדמות.
- getLevel – מחזירה את הרמה.
- getForce – מחזירה את כמות נקודות הכוח.
- getHealthPoints – מחזירה את כמות החיים הנוכחית.
- getCoins – מחזירה את כמות המטבעות.
- getDesctiption – מחזירה מחרוזת המתארת את הדמות לפי הפורמט הבא:

<Name>, *<Job>* with *<Character>* character (level *<Level>*, force *<Force>*)

לדוגמה:

Grendel, Magician with Responsible character (level 8, force 18)

Event 5.1.3

מחלקה זו מייצגת אירוע במשחק. הממשק המחייב של המחלקה כולל מתודה אחת:

- getDescription – מחזירה את התיאור של האירוע.

אם האירוע הינו אירוע מיוחד, התיאור שלו הוא השם של האירוע כפי שנקלט מקובץ הקלט.
אם האירוע הוא מסוג היתקלות במפלצת (פרט ל-Pack) התיאור הינו לפי הפורמט הבא:

<*MonsterName*> (power <*Power*>, loot <*Loot*>, damage <*Damage*>)

לדוגמה, עבור Snail:

Snail (power 5, loot 2, damage 10)

אם האירוע הוא מסוג התקלות עם להקה, התיאור הינו לפני הפורמט הבא:

Pack of <*Size*> members (power <*Power*>, loot <*Loot*>, damage <*Damage*>)

דוגמה ללהקה אפשרית:

Pack of 2 members (power 24, loot 10, damage 50)

במידה ומדובר על להקה שבה יש תת-להקה, לדוגמה הלהקה שמופיעה בקובץ האירועים:

Pack 2 Slime Pack 2 Slime Slime

נדפיס את מספר חברי הלהקה וכל תתי הלהקות שלה ואת פרטי אוסף הלהקות ביחד:

Pack of 3 members (power 36, loot 15, damage 75)

עוד דוגמה כזו:

Input: Pack 2 Slime Pack 2 Slime Pack 2 Slime Slime

Output: Pack of 4 members (power 48, loot 20, damage 100)

5.2 קלט

המערכת מקבלת את מידע על האירועים ועל השחקנים משני קבצים הניתנים על ידי המשתמש כארגומנטים בשורת הפקודה של התכנית. בחלק זה נפרט את הפורמט של כל אחד מהקבצים הללו, תוכלו למצוא דוגמאות נוספות בקבצים המסופקים לתרגיל.

5.2.1 קובץ האירועים

קובץ זה מכיל את רשימת האירועים שהדמויות יחוו במשחק, לפי סדר. בקובץ רשומות מחרוזות מופרדות ברווחים ובירידות שורה (ולעיתים גם מספרים), כך שכל מחרוזת היא שם של אחד האירועים האפשריים שהגדרנו. חייבים להיות ברשימה לפחות שני אירועים.
מקרה מיוחד בקליטה של האירועים הוא קליטת Pack. לאחר שמופיעה בקובץ המחרוזת "Pack" יופיע מספר המעיד על גודל הלהקה. אם גודל הלהקה הוא k, אזי k המחרוזות שיופיעו לאחר מכן הינם המפלצות השייכות ללהקה.
לדוגמה, תוכן אפשרי של קובץ האירועים:

Snail
Pack 2 Slime Slime
Balrog
SolarEclipse
PotionsMerchant

במקרה הזה, האירוע הראשון הוא היתקלות עם Snail, השני הוא היתקלות בלהקה המכילה שני Slime והשלישי הוא התקלות ב-Balrog. לאחר מכן מתרחשים ליקוי חמה ומפגש עם סוחר שיקויים. לאחר המפגש עם סוחר השיקויים חוזרים האירוע הראשון, השני, וכן הלאה... עד סוף המשחק.

בכל מקרה של שגיאה בקריאת הקובץ (למשל אם הקובץ בפורמט לא תקין) יש לזרוק חריגה המכילה את ההודעה "Invalid Events File".

שימו לב, הקובץ הבא חוקי לגמרי כי יש בו רווחים וירידות שורה במקומות רלוונטים:

Snail
Pack 2 Slime
Slime
Balrog

הקובץ הזה מתאר חילזון, אחריו להקה של שני סליימים, ואז באלור.

5.2.2 קובץ השחקנים

קובץ זה מכיל את רשימת השחקנים במשחק לפי הסדר בו ישחקו. חייבים להיות לפחות שני שחקנים במשחק ולא יותר משישה שחקנים. כל שחקן בקובץ מוגדר על ידי שלוש מחרוזות מופרדות ברווח לפי הפורמט הבא:

<Name> <Job> <Character>

- Name – שם הדמות. מחרוזת באורך 3 עד 15 ספרות (כולל).
- Job – העבודה של הדמות.
- Character – אופי הדמות.

לדוגמה, תוכן אפשרי של קובץ שחקנים:

Daniel Warrior Responsible
Regev Magician RiskTaking
Baraa Archer RiskTaking

בכל מקרה של שגיאה בקריאת הקובץ (למשל אם הקובץ בפורמט לא תקין) יש לזרוק חריגה המכילה את ההודעה "Invalid Players File".

5.3 הוראות מימוש

- בתרגיל זה מותר ואף מומלץ להשתמש בכל מה שלמדנו במהלך הסמסטר, כולל מבני נתונים מ-STL, ספריית האלגוריתמים מ-STL, מצביעים חכמים ועוד.
- מותר ואף צריך לשנות את הקבצים המסופקים לכם, פרט לקבצים main.cpp, Utilities.cpp ו-Utilities.h אותם אתם לא מגישים.
- בקובץ MatamStory.cpp המסופק לכם קיימת תבנית למימוש המחלקה. אין חובה לעקוב אחריה, זוהי רק המלצה שלנו, מותר לממש את המחלקה איך שתראו לנכון. כדי לעבוד עם התבנית פשוט מלאו את החלקים החסרים בין ההיערות, היכן שרשמנו "TODO".
- אסור להשתמש בדברים שלא למדנו בהרצאות או בתרגולים.
- ניתן לחלק את התכנית שלכם לכמה קבצים שתראו לנכון. על הקבצים להיות בתיקיה הראשית של הפרויקט, בתיקיה Players או בתיקיה Events בלבד.
- מותר לכם להגדיר ולממש מספר מחלקות באותו קובץ רק אם הקובץ נשאר יחסית קטן ויש קשר הגיוני בין המחלקות.

5.4 פלט התכנית

בקובץ Utilities.h המסופק לכם תוכלו למצוא מגוון פונקציות הדפסה לשימושכם המתאים לכל צרכי הפלט לאורך המשחק. ההדפסות של הפונקציות הללו תואמות את הפורמט שהבדיקות האוטומטיות מצפות לו, לכן כדאי להשתמש רק בפונקציות ההדפסה המסופקות ולא להדפיס לערוצי הפלט בעצמכם כלל.
בקובץ Utilities.h תוכלו למצוא תיעוד של הפונקציות השונות, מומלץ לקרוא את התיעוד כדי שתוכלו להשתמש בהן כהלכה.
הודעות ההדפסה שמתוארות בקובץ ובהמשך הקובץ הזה הן חשובות מאוד ומאפשרות לעשות סטנדרטיזציה של הפלט שלכם, ומאפשרות לנו לבדוק לכולכם את התרגיל בצורה אוטומטית. חשוב שתשתמשו בהן, לא נקבל ערעורים שנובעים מפלט שגוי כי לא השתמשתם בהדפסות שלנו. על כל פלט שלא הגיע מפונקציית פלט שלנו, תשאלו את עצמכם "האם יש פונקציה קיימת שאני אמור לפלוט איתה במקום?".

5.4.1 הדפסת הודעת פתיחה

בתחילת המשחק, לאחר שהאירועים והשחקנים נקלטו מהקבצים, תודפס הודעת לפתיחת המשחק. בהודעה זו יפורטו השחקנים המשתתפים במשחק לפי הסדר בו הם משחקים.

את תחילת ההודעה יש להדפיס עם הפונקציה printStartMessage. לאחר מכן, עבור כל אחד מהשחקנים המשתתפים, יש להדפיס את הפירוט על השחקן באמצעות הפונקציה printStartPlayerEntry. לאחר שהדפסנו את כל השחקנים, יש לסיים את הודעת ההתחלה בהדפסת חוצץ באמצעות קריאה ל-printBarrier.

5.4.2 הדפסת לאורך תור

לפני ביצוע התור, תודפס הודעה המפרטת על השחקן ועל האירוע המשוחק באמצעות הפונקציה printTurnDetails. לאחר ביצוע התור, תודפס הודעה המעידה על תוצאת התור באמצעות הפונקציה printTurnOutCome.

שימו לב שהפונקציה מקבלת את התוצאה (outcome) של התור כארגומנט, ניתן לקבל את הודעת התוצאה המתאימה באמצעות הפונקציות הבאות בהתאם לסוג האירוע:

- getEncounterWonMessage – למקרה בו הדמות נתקלה במפלצת וניצחה.
- getEncounterLostMessage – למקרה בו הדמות נתקלה במפלצת והפסידה.
- getMerchantPurchaseMessage – למקרה בו הדמות פגשה סוחר שיקויים
- getSolarEclipseMessage – למקרה בו הדמות חוותה ליקוי חמה.

5.4.3 הדפסות לאורך סיבוב

בתחילת כל סיבוב, תודפס הודעה המעידה שמתחיל סיבוב חדש על ידי הפונקציה printRoundStart. לאחר הדפסת הסיכומים של כל התורות בסיבוב, תודפס הודעת printRoundEnd.

לבסוף תודפס טבלת הדירוג העדכנית – תחילה תודפס הודעה להתחלת הטבלה באמצעות printLeaderBoardMessage ולאחר מכן יודפסו כל הדמויות לפי סדר הדירוג באמצעות printLeaderBoardEntry. בסוף הטבלה יודפס חוצץ באמצעות printBarrier.

5.4.4 הדפסות בסוף המשחק

כאשר המשחק מסתיים, מודפסת הודעת סיום באמצעות printGameOver.
אם אחת הדמויות ניצחה – תודפס הודעה באמצעות printWinner.
אם אין מנצח במשחק – תודפס הודעה באמצעות printNoWinner.

5.5 הידור ובדיקות מסופקות

יש לקמפל את הבדיקות עם הקבצים שכתבתם באמצעות הרצת הפקודה הבאה מהתיקיה הראשית של הפרויקט (ללא ירידת שורה):

```
g++ --std=c++17 -o MatamStory -Wall -pedantic-errors -Werror -DNDEBUG *.cpp
Events/*.cpp Players/*.cpp -I. -I./Players -I./Events
```

בתיקיה tests תוכלו למצוא מספר קבצי קלט לדוגמה ופלטים מתאימים לכל קלט. ניתן להריץ את התכנית עם הקלטים המסופקים למשל באמצעות השורה הבאה:

```
./MatamStory tests/test1.events tests/test1.players > test1.out
```

תוכלו לבדוק את פלט התכנית אל מול הפלט המצופה למשל באמצעות הפקודה הבאה:

```
diff --strip-trailing-cr -B -Z test1.out tests/test1.expected
```

6 הגשה ובדיקה

6.1 הגשה

עליכם להגיש את הקבצים הבאים:

- dry.pdf
- log.txt
- MatamStory.h
- MatamStory.cpp

בנוסף, מותר לכם להגיש קבצים עם סיומת h. או סיומת cpp. בתיקיות Events ו-Players בלבד.

את ההגשה יש לבצע במערכת ה-Gradescope.

שימו לב – על אחד השותפים (בלבד) להגיש את פתרון התרגיל במערכת. לאחר ההגשה יש להוסיף את השותף השני להגשה באמצעות לחיצה על כפתור "Group Members" ב-Gradescope.

הערות:

- ניתן להגיש מספר פעמים את התרגיל, ההגשה האחרונה (בלבד) היא זו שתיבדק ותקבל ציון.
- בכל הגשה נוספת שאתם מבצעים, יש להוסיף מחדש את השותף להגשה ב-Gradescope.
- וודאו שהקבצים אותם אתם מגישים (בפרט קבצי PDF) ניתנים לפתיחה ולצפיה.

6.1.1 הגשה באמצעות GitHub

בתרגיל זה, יש להגיש את הפתרון שלכם במערכת ה-Gradescope באמצעות GitHub. עליכם להגיש את ה-Repo שיצרתם עבור תרגיל זה (כפי שיצרתם בתרגילים הקודמים) אליו הוספתם את הפתרון לחלק היבש ולחלק הרטוב. לשם כך בדף הגשת התרגיל בחרו באפשרות ההגשה באמצעות GitHub, תנו ל-Gradescope הרשאות גישה לחשבון ה-GitHub שלכם ובחרו את ה-Repo אותו תרצו להגיש.

הקפידו על עבודה נכונה עם Git כפי שלמדנו בהרצאות ובתרגולים.

6.2 בדיקה אוטומטית

מיד עם ההגשה שלכם במערכת ה-Gradescope, יתבצעו מספר בדיקות אוטומטיות על קובץ ההגשה:

1. בדיקות שפיות – בודקות שכל הקבצים הדרושים נמצאים ב-zip המוגש ושהקוד שלכם מתקמפל כראוי. בדיקות אלו נועדו לוודא שפורמט ההגשה שלכם תקין.
2. טסטים אוטומטיים – הבדיקה האוטומטית מריצה את התכניות שהגשתם עם הקלטים שסופקו לכם בקבצי התרגיל. הטסטים האוטומטיים בודקים שריצת התכנית הסתיימה בהצלחה ללא שגיאות זכרון ונתנה פלט נכון.

בדיקות אלו נועדו עבורכם לוודא שההגשה שלכם תקינה והקוד עובד כראוי, קראו היטב את הפלט והמשוב מהבדיקות!

על מנת להשוות תכנים של קבצים בבדיקה האוטומטית, נשתמש בפקודת ה-diff הבאה (כאשר file1 ו-file2 הם הקבצים אותם נרצה להשוות:

```
diff --strip-trailing-cr -B -Z file1 file2
```

<u>שימו לב</u> – בנוסף לבדיקות שסופקו לכם, ההגשה הולכת לעבור בדיקות אוטומטיות נוספות לפיהן יקבע הציון על התרגיל. אנו מצפים שתבדקו בעצמכם את התרגיל שלכם מעבר לבדיקות המסופקות, האחריות על נכונות ההגשה היא שלכם בלבד.

6.3 תחרות מימז

בתרגיל זה מתקיימת תחרות מימז (Memes).
כדי להשתתף בתחרות, עליכם להוסיף מימ לתחילת הקובץ dry.pdf. הזוג שיזכה בתחרות יקבל בונוס של 5 נקודות לציון התרגיל (לכל אחד מהשותפים).

חוקי התחרות:

1. מותר לכם להגיש לתחרות מימ אחד <u>בלבד</u>.
2. נושא המימ חייב להיות קשור לחומר הקורס.
3. המימ הזוכה יפורסם בתרגול לאחר לפרסום הציונים.

בהצלחה !
بالنجاح !
GOOD LUCK!