רשתות תקשורת מחשבים
סמסטר א' תשפ"ה

תרגיל בית מס' 3.

להגשה עד 11.01.2025.

חלק ראשון – UDP-Cman (65%):

בחלק זה של התרגיל נממש גרסה של משחק המוכר והאהוב "Cman". לתוכנה יהיה צד שרת וצד לקוח שיתקשרו ב-UDP. **הלוגיקה של ניהול המשחק עצמו מומשה עבורכם, עליכם יהיה לממש את השרת שיריץ אותה, ואת הלקוחות.**

תוכנת הלקוח תתחבר לשרת ותבקש ממנו לשחק בתור Cman או בתור הרוח, או להתחבר כצופה מהצד. המשחק מתחיל ברגע שכל תפקיד שחקן (Cman והרוח) מאויש.

מהרגע שהמשחק מתחיל, תוכנת הלקוח תדפיס את מצב המשחק הנוכחי לטרמינל לפי המידע שקיבלה מהשרת, תאזין ללחיצות המקשים של הלקוח ותשלח לשרת פקודות תזוזה (מעלה/שמאלה/מטה/ימינה) או יציאה מהמשחק בהתאם לאותן לחיצות.

חוקי המשחק:

המפה בנויה מרשת מלבנית של משבצות. כל משבצת יכולה להיות פנויה או תפוסה ע"י קיר. 40 מהמשבצות הפנויות מתחילות עם נקודות עליהן. שתי משבצות פנויות אחרות מסווגות כנקודות ההתחלה של כל אחד מהשחקנים.

השחקנים יכולים לנוע מעלה, שמאלה, מטה או ימינה בין משבצות פנויות, משבצת אחת בכל פעם.

Cman יכול להתחיל לזוז מייד עם תחילת המשחק, הרוח יכולה להתחיל לזוז רק לאחר ש-Cman התחיל לזוז.

Cman יכול לאסוף את הנקודות ע"י דריכה על המשבצת שלהן, והוא צריך לאסוף 32 מהן בשביל לנצח.

הרוח יכולה לתפוס את Cman ע"י דריכה על המשבצת עליה הוא עומד, ותנצח ברגע שהוא ייתפס 3 פעמים.

אם Cman נתפס, הוא והרוח חוזרים לנקודת ההתחלה שלהם, והגבלות התנועה על הרוח מתחדשות (הניקוד נשמר ונקודות שנאספו לא חוזרות.)

מבנה המפה

בקובץ הטקסטואלי ממנו המפה נטענת, התו **F** מייצג משבצת פנויה, **W** מייצג משבצת קיר, **P** מייצג משבצת עם נקודה, **C** מייצג את נקודת ההתחלה של Cman ו-**S** מייצג את נקודת ההתחלה של הרוח.

בקובץ ה-utils המצורף יש פונקציה read_map שטוענת את קובץ המפה, מוודאת שהוא תקני ומחזירה אותו כמחרוזת.

למי שרוצה לבנות מפות משלו בשביל הכיף, יש לדאוג שכל השורות שוות אורך, שכל המשבצות החיצוניות הן קירות, ושיש בדיוק נקודת התחלה אחת לכל שחקן ו-40 נקודות.
גודל המפה המקסימלי הוא 255×255, כולל קירות המסגרת.

אופי הפעולה:

שרת:

השרת ישתמש ב-UDP socket על מנת לתקשר עם הלקוחות.

כשלקוח חדש יבקש להצטרף בתפקיד שעוד לא נבחר, או בתור צופה, השרת ירשום אותו לתפקיד וישלח לו הודעת עדכון עם המידע על מצב המשחק. אם התפקיד כבר תפוס (יכולים להיות מספר צופים, אבל רק שחקן אחד בכל תפקיד), השרת ישלח לו הודעת שגיאה. אם לקוח שולח הודעת יציאה לפני תחילת המשחק, השרת יפנה את התפקיד ויחכה שלקוח אחר ייקח אותו.

ברגע שכל תפקידי השחקנים יאוישו, השרת יאפשר את תחילת המשחק. השרת יחכה להודעות מהלקוחות, יעדכן את מצב המשחק וישלח ללקוחות מידע על מצב הלוח העדכני ביותר. אם משתמשים יבקשו להצטרף בתור שחקנים נוספים בשלב הזה, הם ייענו בסירוב בהודעת שגיאה, אבל צופים יוכלו להצטרף באמצע ולצפות במשחק מהרגע שבו הצטרפו.

ברגע שאחד מהשחקנים ישיג את תנאי הניצחון שלו, כולם יקבלו הודעה שאותו שחקן ניצח, ולאחר 10 שניות השרת יתחיל משחק חדש וימתין ללקוחות חדשים (בזמן הזה השרת ימשיך לשלוח את תוצאות המשחק אחת לשנייה למקרה שהודעות נפלו בדרך!

אם אחד מהשחקנים ייצא באמצע המשחק, השחקן השני ינצח באופן אוטומטי.

הלקוח:

כשהלקוח יתחבר, הוא יקבל מהשרת את המידע על מצב המשחק הנוכחי, ואם הוא שחקן הוא יחכה לקבל "יריית זינוק" (אישור תנועה) מהשרת (צופה מהצד לעולם לא יקבל יריית זינוק, מן הסתם.)

לאורך כל זמן המשחק, הלקוח (בין אם הוא שחקן או צופה) צריך לבדוק* כל הזמן האם המקש Q נלחץ, ואם כן אז לשלוח לשרת הודעת התנתקות ולצאת.

בנוסף, לאורך כל זמן המשחק, הלקוח (בין אם הוא שחקן או צופה) צריך לבדוק כל הזמן האם השרת שלח עדכונים, ואם כן, לעדכן את מצב הלוח שמודפס למסך בהתאם לעדכונים.*

אחרי שיריית הזינוק מתקבלת מהשרת, תוכנת הלקוח צריכה גם לבדוק* כל הזמן האם מקשי התזוזה (WASD, W=מעלה, A=שמאלה, S=מטה, D=ימינה) לחוצים, ואם כן אז לשלוח לשרת הודעה עם הכיוון אליו הוא זז (לחיצה אחת = צעד אחד. אם כמה לחוצים בו זמנית אתם יכולים להחליט בעצמכם מי יקבל עדיפות.)

כשמתקבלת הודעת סוף משחק מהשרת, התוצאות הסופיות יודפסו למסך והתוכנה תצא.

בכל שלב, אם התוכנה תקבל הודעת שגיאה מהשרת היא תדפיס הודעת שגיאה מתאימה ותצא.

***לנוחיותכם, בקובץ ה-utils המצורף יש מימוש לפונקציה get_pressed_keys שמחזירה את רשימת המקשים הלחוצים ברגע נתון בעזרת ספריית pynput ולפונקציה clear_print שמנקה את המסך לפני הדפסה (לצורך עדכון מצב הלוח).**

הרצה:

מומלץ להשתמש בספרייה argparse בשביל לקבל את הפרמטרים.

שורת ההרצה של השרת:

python cman_server.py -p port

- -p port
  פרמטר אופציונלי המסמן את מספר הפורט הנבחר. ערך ברירת המחדל יהיה 1337.

שורת ההרצה של הלקוח:

python cman_client.py role addr -p port

- role
  פרמטר חובה שיכיל אל התפקיד שהלקוח רוצה לשחק (cman, spirit או watcher).
- addr
  פרמטר חובה שיכיל את כתובת השרת. יכול להיות כתובת IP או hostname.
- -p port
  פרמטר אופציונלי שמכיל את מספר הפורט הנבחר. ערך ברירת המחדל יהיה 1337.

הדפסות:

לצורך הדפסת מצב הלוח הנוכחי, אתם יכולים להניח שקובץ המפה נגיש לכל המשתמשים כמו גם לשרת, ולכן כפי שתראו בהמשך השרת צריך לשלוח את כל הלוח בכל הודעה אלא רק מידע על דברים שמשתנים במהלך המשחק שבאמצעותו ובאמצעות קובץ המפה המקומי ניתן לשחזר את מצב הלוח.

אתם יכולים להחליט בעצמכם כיצד להדפיס למסך את מצב המשחק (באילו תווים לייצג קירות, נקודות, שחקנים וכו') כל עוד הכל נוח וברור למשתמש אנושי מה אמור להיות מה, וכל עוד הכל בתוך הטרמינל.

הודעות הפרוטוקול:

כל הודעה מתחילה בבית אחד של OPCODE ייחודי לסוג ההודעה:

Direction	OPCODE Value	Name
Client->Server	**0x00**	Join
Client->Server	**0x01**	Player movement
Client->Server	**0x0F**	Quit
Server->Client	**0x80**	Game state update
Server->Client	**0x8F**	Game end
Server->Client	**0xFF**	Error

מבנה ההודעה נקבע לפי ה-OPCODE כפי שמתואר להלן:

בקשת הצטרפות:

- **OPCODE**
- **ROLE** – בית אחד (0=צופה, 1=Cman, 2=רוח)

הודעת תזוזת שחקן:

- **OPCODE**
- **DIRECTION** – בית אחד (0=מעלה, 1=שמאלה, 2=מטה, 3=ימינה)

הודעת יציאה:

- **OPCODE**

עדכון מצב משחק:

- **OPCODE**
- **FREEZE** – בית אחד שערכו 0 אם הנמען יכול לשלוח הודעות תזוזה ו-1 אם לא (למשל: עוד לא התחבר שחקן שני \ המשתמש משחק את הרוח ו-Cman עוד לא זז \ המשתמש רק צופה.) ההודעה הראשונה בכל סיבוב עם 0 בשדה זה היא "יריית הזינוק" לאותו שחקן.
- **C_COORDS** – שני בתים המייצגים את הקואורדינטות של Cman (בית אחד לכל ציר) או FFFF אם אף שחקן עוד לא התחבר בתור Cman.
- **S_COORDS** – שני בתים המייצגים את הקואורדינטות של הרוח (בית אחד לכל ציר) או FFFF אם אף שחקן עוד לא התחבר בתור הרוח.
- **ATTEMPTS** – בית אחד המייצג את מספר הפעמים ש-Cman נתפס ע"י הרוח עד כה.
- **COLLECTED** – חמישה בתים, כל ביט מייצג את אחד מ-40 הנקודות הפזורות על הלוח, וערכו 0 אם היא עדיין נוכחת או 1 אם Cman אסף אותה (אופן ההתאמה בין נקודות לביטים נתון לבחירתכם.)

סוף משחק:

- **OPCODE**
- **WINNER** – בית אחד שערכו הוא 1 אם Cman ניצח או 2 אם הרוח ניצחה.
- **S_SCORE** – בית אחד המכיל את מספר הפעמים שהרוח תפסה את Cman.
- **C_SCORE** – בית אחד המכיל את מספר הנקודות ש-Cman אסף.

שגיאה:

- **OPCODE**
- **DATA** – 11 בתים של סוג שגיאה + כל מידע נוסף שאתם רואים לנכון, בהתאם לצרכים שלכם.

רשתות תקשורת מחשבים
סמסטר א' תשפ"ה

חלק שני – ICMP, כתובות IP, ואלגוריתמי ניתוב ברשת (35%):

שאלה 1 – ICMP (5%):

הסבירו בקצרה כיצד ניתן להשתמש ב-ICMP על מנת לגלות אילו נתבים נמצאים על המסלול מהמחשב שלנו למחשב אחר.

שאלה 2 - כתובות IP (15%):

רשת כלשהי מחולקת למספר תת רשתות. נתב בארגון מכיל את טבלת הניתוב הבאה:

Network Name	Network	Exit port
A	192.168.1.128/25	1
B	192.168.2.0/23	2
C	192.168.1.128/26	2
D	192.168.1.160/27	1
E	192.168.3.128/25	2
F	0.0.0.0/0	3

א. מה התפקיד של שורה F?

ב. חשבו עבור 5 החבילות הבאות לאילו יציאות הן ינותבו:

Datagram Name	Dest IP Address
P	192.168.1.21
Q	192.168.1.201
R	192.168.2.1
S	192.168.3.0
T	192.168.3.255

ג. האם ניתן למחוק שורה מהטבלה בלי לשנות את הניתוב של אף הודעה (באופן כללי, לא רק החמש מהסעיף הקודם)?

רשתות תקשורת מחשבים
סמסטר א' תשפ"ה

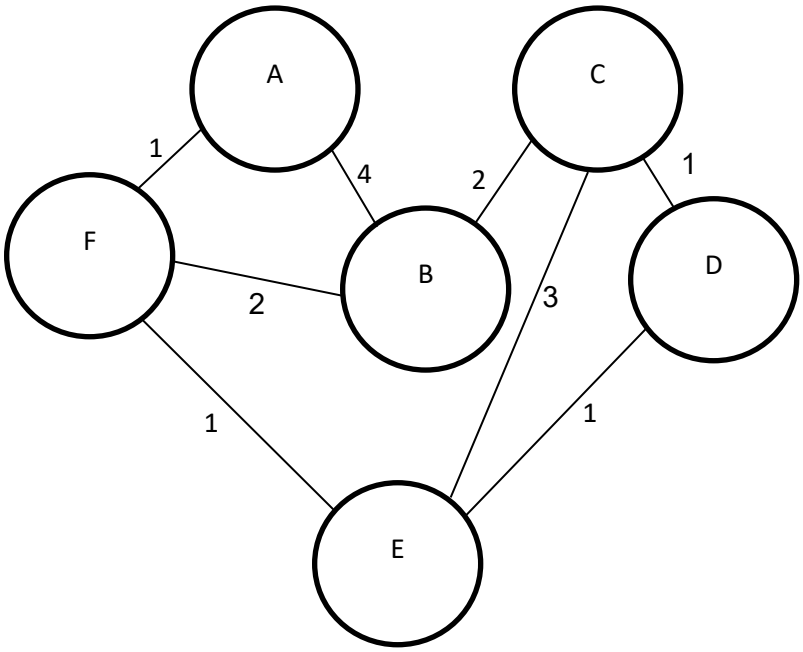
שאלה 3 – אלגוריתמי ניתוב (15%):

נתונה הרשת באיור למטה, כולל עלויות הקשתות בין הצמתים השונים.

א. נניח והרשת משתמשת באלגוריתם הDistance Vector למציאת מסלולי ניתוב ברשת, בצורתו הפשוטה. מהי טבלת הניתוב בצומת A (כלומר – מרחקיו מכל שאר הצמתים ברשת) בתחילת התהליך? ובסיום התהליך (לאחר התייצבות האלגוריתם)? הציגו גם את שדה הnext hop.

ב. נניח כעת כי נותקה הקשת E-F. האם האלגוריתם יתייצב? אם כן, מה יהיו כעת המרחקים בצומת A? אם לא, כיצד תראה טבלת המרחקים בצומת D לאורך הזמן?

ג. ענו שוב על הסעיף הקודם עבור המקרה שבו לאחר המקרה שתואר בסעיף סעיף ב' נותקה גם הקשת C-B.



הנחיות הגשה (לתרגיל בשלמותו):

- הגישו את התרגיל כקובץ ZIP אחד, המכיל בתוכו:
  (1) קובץ ZIP נוסף עבור החלק הראשון (המעשי) של התרגיל, שיכיל את קבצי הקוד.
  (2) PDF עם התשובות לחלק התאורטי.

- לZIP הראשי קראו ID.zip אם הגשתם את התרגיל לבד, או ID1_ID2.zip אם הגשתם בזוג.