בשאלה זו נכתוב מערכת שתסמלץ קרב בין שני צבאות. אנחנו נכתוב את התוכנה הזו תוך שימוש בפונקציות ותכנון top down.

שדה הקרב יהיה לוח מלבני, כאשר **כל משבצת בלוח מייצגת שטח ריבועי שצלעו היא 5 מטרים.**

שלב ראשון - אתחול הקרב

בתחילת הקרב יוכנסו לתוכנה פרטים על מספר המשבצות לאורך ולרוחב בשדה הקרב. ניתן להניח שערכי האורך והרוחב לא עולים על 15 משבצות (אך אין הכרח שיהיו שווים).

לאחר מכן, יוכנס לתוכנה המצב ההתחלתי של לוח הקרב. כלומר יוכנס אליה מערך של מספרים התואם בגודלו את הגודל שהוכנס בשלב הקודם. במערך מספרים זה "0" מייצג תא ריק, מספר חיובי מייצג חייל של "צבא החיוביים" עם "חוזק" שזהה למספר, מספר שלילי מייצג חייל של "צבא המספרים השליליים" עם "חוזק" שזהה למספר כפול 1-. ניתן להניח שהערך המוחלט של כל עצמה הוא 9999 לכל היותר**ניתן להניח שכל הקלט חוקי**

דוגמאות לקלט:

- הקלט

1 1

1234

מייצג שדה קרב של תא אחד. בתא הזה יש חייל של "צבא החיוביים", כאשר עצמתו היא 1234

- הקלט

2 3

0 20 0

-10 0 -10

	20	
10		10

מייצג את שדה הקרב הנ"ל, כאשר חיילים חיוביים מסומנים באדום וחיילים שליליים מסומנים בכחול.

כלומר, מייצג שדה קרב של שתי שורות ושלוש עמודות שבו יש חייל של "צבא החיוביים" בתא השני של השורה הראשונה עם עצמה 20, וחיילים של "צבא השליליים" בתאים הראשון והשלישי של השורה השניה, לכל אחד מהם עצמה 10.

שלב שני - ניהול הקרב

לאחר שקלטנו את המידע בהצלחה, התוכנה תסמלץ את מהלך הקרב. אופן הסמלוץ מוגדר על ידי האלגוריתם הבא:

1. בדיקה האם הקרב הסתיים. הקרב הסתיים אם כל החיילים של צבא מסויים מתו. במקרה כזה התוכנה תדפיס את תוצאת הקרב (Positive win, Negative win) ואת המצב הסופי של לוח הקרב, ותצא. הסבר מפורט יותר על פורמט הפלט מופיע בהמשך.

2. בחירת החייל שיפעל כרגע:

   a. **שלב מציאת סדר פעולת החיילים:** ראשית, עוברים על כל שדה הקרב (מהשורה העליונה לתחתונה, ובכל שורה משמאל לימין). כל חייל יפעל לפי הסדר שנתקלנו בו במעבר הזה. כלומר: ראשית יפעל החייל שנמצא בפינה השמאלית העליונה (אם קיים), לאחר מכן החייל שמימינו (אם קיים) וכו.

   b. מבצעים את שלב 3 (שלב פעולת החייל) פעם אחת על כל אחד מהחיילים לפי הסדר ששלב מציאת הסדר בחר, או עד שהקרב הסתיים. לאחר שביצענו את שלב 3 פעם אחת על כל חייל, מריצים את שלב מציאת הסדר שוב כדי לבדוק האם הסדר בין החיילים השתנה (החיילים יכולים לזוז על הלוח, דבר שישנה את סדר פעולתם בסיבוב הבא).

3. **שלב פעולת החייל**: בחירת הפעולה של החייל שפועל כרגע (החיילים תומכים בשתי פעולות: "התקפה" ו"תזוזה")

   a. **שלב מציאת האויב הקרוב ביותר:** תחילה, חייל ימצא את חייל היריב הקרוב אליו ביותר. אם יש כמה חיילי יריב באותו המרחק, הוא יבחר את החייל שנמצא **בשורה התחתונה ביותר**. אם יש כמה חיילי יריב קרובים ביותר באותה השורה, ייבחר החייל **בטור השמאלי** ביותר מביניהם.

   b. **התקפת האויב הקרוב ביותר, אם ניתן:** אם חייל היריב הקרוב ביותר נמצא במרחק של 10 מטרים לכל היותר, אז הוא מותקף. במקרה כזה, מורידים 10 מעוצמת חייל היריב שהותקף, והפעולה של החייל התוקף מסתיימת. אם החייל שהותקף ירד לעוצמה 0 אז הוא נחשב כמת ולא יוכל לפעול יותר בהמשך הקרב.
   שימו לב: עוצמתו של חייל לא יכולה להיות שלילית!

   c. **אחרת, תזוזה:** אם כל חיילי היריב רחוקים מדי, החייל שלנו ינסה לזוז. החייל יזוז למשבצת פנויה בשדה הקרב, שנמצאת במרחק של 25 מטרים לכל היותר ממנו, ושתקרב אותו למרחק מינימלי מחייל האויב הכי קרוב אליו (זה שנמצא בשלב מציאת האויב הקרוב ביותר). במידה ואין משבצת כזו (אין משבצת פנויה במרחק של 25 מטרים או שכל המשבצות הפנויות רק ירחיקו אותנו מחייל האויב שקרוב ביותר כרגע, או ישאירו אותנו באותו המרחק), החייל לא עושה כלום ותורו מסתיים.

   במידה ויש כמה משבצות פנויות שיקרבו אותו למרחק המינימלי מחייל האוייב הקרוב ביותר, החייל יזוז **למשבצת העליונה ביותר** שמקיימת את התנאי. במידה ויש כמה

משבצות כאלו באותה השורה, החייל יזוז **למשבצת הימנית ביותר** שמקיימת את התנאי.

שלב שלישי – הדפסת תוצאת הקרב. פורמט הפלט:

התוכנה פולטת פלט רק כאשר הקרב מסתיים. ראשית מודפסת מחרוזת שמצביעה על זהות המנצח. ניתן להניח שאין תיקו (כלומר אין שדה קרב ריק), ולכן המחרוזת היא אחת מבין המחרוזות הבאות:

- Positive win! (אם יש רק "חיילים חיוביים" בשדה הקרב).
- Negative win! (אם יש רק "חיילים שליליים" בשדה הקרב)

לאחר מכן התוכנה תדפיס את שדב הקרב בפורמט הבא:

- בין כל שני תאים יופיע התו "|".
- על המספר שבכל תא להיות מיושר ל4 תווים. כלומר – אם צריך להדפיס מספר בן ספרה אחת, הדפיסו גם שלושה רווחים לפניו. (ניתן להניח שלא תצטרכו להדפיס מספר עם יותר מ4 ספרות).
- **ראו דוגמאות קונקרטיות בקבצי הדוגמאות.**
  בכל מקרה, כל החלקים שאחראים על פלט התוכנה כבר מולאו בשבילכם בקובץ הskeleton, אתם לא צריכים לממש אותם מחדש.

דוגמאות מפורטות:

לשרותכם כ5 דוגמאות מפורטות. מטרתן היא גם לתת לכם קלט ופלט לדוגמה, אך גם לעזור בהבהרת הדרישות בתרגיל. עבור כל דוגמת הרצה מסופקים לכם 4 קבצים:

- קובץ קלט המכיל קלט לדוגמה והנמצא בתיקיה inputs תחת tests.zip.
- קובץ פלט המכיל את הפלט המתאים לקלט לדוגמה. **זה הפלט עימו יש להשוות את פלט הפתרון שלכם** לפני ההגשה. נמצא בתיקיה outputs תחת tests.zip
- detailed_output.txt שמכיל פלט כאשר פולטים את מצב הלוח לאחר כל תזוזה. **פלט זה לא מייצג את מה שאמור להיות מוגש בתרגיל!**, והוא שם כדי לעזור לכם להבין ולמצוא באגים בתוכנית שלכם במהלך כתיבת הקוד (כלומר: אם אחת הבדיקות לא עוברת ואינכם מבינים למה, תוכלו להוסיף הדפסות נוספות, להשוות מול detailed_output.txt ולהבין מהו "שלב הביניים" שבו סטיתם מהפתרון הנכון). **נמצא בתיקיה explanations תחת tests.zip.**
- example_explained.txt –**כולל הסברים מפורטים בעברית על הפלט של detailed_output שמטרתם להבהיר סופית את דרישות התרגיל. הנכם מתבקשים לעבור על הדוגמאות ובייחוד על הקבצים האלו בעיון רב על מנת להמנע מאי בהירות מיותרת בנוגע לדרישות התרגיל. נמצא גם הוא בתיקיה explanations תחת tests.zip.**

להלן תיאור הפונקציות שירכיבו את פתרון התרגיל. שימו לב שיש כאן חלוקה להרבה פונקציות, אך כל פונקציה יכולה להיות קצרה יחסית (הפתרון הרשמי לתרגיל יכלול פחות מ200 שורות קוד, מתוכן 100 כבר סופקו לכם). כל פונקציה יכולה להשתמש בפונקציות שיש לפניה ברשימה.

---

void initBattle(int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int *, int *)

פונקציה זו אחראית על קליטת שדה הקרב מהמשתמש. היא תקלוט כשני מספרים שמייצגים את מספר מספר השורות ומספר העמודות בשדה הקרב , ולאחר מכן תמלא את שדה הקרב במספרים שמוכנסים על ידי המשתמש.

---

void initBattleActionOrder (int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int, int, int *)

פונקציה זו אחראית על קביעת סדר החיילים שיפעלו להבא. (כפי שמתואר בסעיף מציאת סדר פעולת החיילים בהסבר על ניהול הקרב). הפונקציה תקבל את שדה הקרב כפרמטר הראשון, את מספר השורות והעמודות האמיתיים של הקרב בפרמטר השני והשלישי, ומערך של מספרים שהפונקציה תמלא כoutput parameter ושיקבע את סדר הפעולות של החיילים.

פונקציה זו ממומשת באופן חלקי בקובץ הskeleton שסופק לכם. יש להשלים אותה באמצעות ההערות המופיעות בקובץ.

---

double getDistance(int , int , int, int)

מקבלת ארבעה פרמטרים שמייצגים שתי נקודות בשדה הקרב (לכל נקודה ניתנים שני פרמטרים – אורך ורוחב), ומחזירה את המרחק בין שתי הנקודות (לפי הנוסחה למרחק בין נקודות בקו ישר).
הפונקציה כבר ממומשת במלואה בקובץ הskeleton שסופק לכם ואין צורך לגעת בה.
שימו לב שאתם לא מוחקים את השורה #include<math.h> מתחילת הskeleton כי זה יגרום לתוכנה שלכם לא להתקמפל.

---

enum BattleResult getBattleResult(int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int , int)

כאשר enum BattleResult מוגדר באופן הבא:

enum BattleResult{POSITIVE_WIN, NEGATIVE_WIN, NO_RESULT};

פונקציה זו תסרוק את שדה הקרב ותחזיר לנו מה מצב הקרב כרגע (POSITIVE_WIN אם נותרו רק מספרים חיוביים במערך, NEGATIVE_WIN אם נותרו רק מספרים שליליים, NO_RESULT אם יש במערך גם מספרים חיוביים וגם שליליים

היא תקבל כפרמטרים את שדה הקרב בפרמטר הראשון, יחד עם אורכו ורוחבו האמיתיים בפרמטרים השני והשלישי.

---

void findClosestEnemy(int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int , int ,int, int, int *, int *)

פונקציה זו אחראית על למצוא את האוייב הקרוב ביותר של חייל, כמו שמפורט בחלק של מציאת האויב הקרוב ביותר בהסבר על ניהול הקרב. פונקציה זו מקבלת כפרמטרים את שדה הקרב, את אורכו ורוחבו האמיתיים, ואת אינדקס השורה והעמודה של החייל שאת האויב הכי קרוב אליו אנחנו צריכים למצוא.

הפונקציה מחזירה את אינדקסי השורה והעמודה של האוייב הקרוב ביותר. **במצביעים**

---

void attack(int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int , int);

פונקציה זו אחראית לעדכון העוצמה החדשה של חייל שעבר מתקפה. היא מקבל כפרמטרים את שדה הקרב, ואת אינדקסי השורה והעמודה של החייל שמותקף ושאת עצמתו צריך לעדכן. שימו לב שעדכון העוצמה מוריד 10 מהעוצמה (כלומר: הורדת המספר של חייל חיובי ב10 עד למינימום של 0 או העלאת המספר של חייל שלילי ב10 עד למקסימום של 0).

---

void move(int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int , int, int, int, int, int);

פונקציה זו אחראית להזזת חייל. היא מקבלת את שדה הקרב, את אורכו ורוחבו האמיתיים, את אינדקסי השורה והעמודה של החייל שצריך להזיז, ואת אינדקסי השורה והעמודה של החייל הקרוב ביותר (כלומר, זה שנרצה שהתזוזה שלנו תביא אותנו למרחק קצר ככל האפשר ממנו).

---

void displayBoard(int [MAX_BATTLE_SIZE][MAX_BATTLE_SIZE], int , int)

פונקציה זו אחראית על הדפסת הלוח (כלומר הדפסת שדה הקרב עם סימני ה| והריווח הנכון כפי שמוסבר בפסקה על פורמט הפלט.

פונקציה זו ממומשת במלואה בקובץ הskeleton שסופק לכם ואין צורך לגעת בה.

פונקציית main שתחבר את כל התוכנה:

- תקלוט את שדה הקרב מהמשתמש באמצעות הפונקציה initBattle
- תנהל את הקרב באמצעות הפונקציות:
  - initBattleOrdering
  - getDistance
  - getBattleResult
  - findClosestEnemy
  - move
  - attack