המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

7.2.2021

פרויקט מסכם מבוא לתכנות

מגישים :

אליאס נסיף , ת.ז: 317721488

עידו בן הרוש, ת.ז: 316439116

עבור :מר שלו יואב וד"ר שאול סלומון

יום ד מעבדה 15:50

חלק א':

תיאור המטלה:
עלינו לכתוב תוכנית למימוש משחק לוח ל6-2 שחקנים, המשחק יכלול לוח ריבועי אשר יהיה בנוי ממשבצות ,וחיילי משחק בצבע שונה לכל משתתף.
בתחילת המטלה עלינו ליצור תרשים זרימה שבו אנו נציג את התוכנית שלנו והשתלשלות העניינים שקורים במהלך המשחק \במהלך כל תור.
במהלך המשחק המשתמש יבחר את מספר השחקנים שהוא יהיה מעוניין לשחק מולם ובנוסף לבחור את גודל הלוח של המשחק (מטריצה ריבועית).
לאחר מכן יוצג לשחקנים הלוח המוכן לתחילת המשחק שבו יוצגו לו כמות החיילים שחושבו עבור כל שחקן ואת הקוביות הצבועות בלוח(כמספר החיילים) עבור כל צבע של כל שחקן.
סידור החיילים על גבי הלוח יהיה באופן אקראי אבל נדאג לזה שאף חייל של שחקן לא ייפול על גבי המשבצת שצבועה בצבע שלו.
בכל תור של כל שחקן יהיו עליו לבחור אחת מ5 האופציות של תזוזת השחקנים על גבי הלוח:
1. הזזת כל החיילים משבצת אחת למעלה (החיילים בשורה העליונה עוברים לשורה התחתונה).
2. הזזת כל החיילים משבצת אחת למטה (החיילים בשורה התחתונה עוברים לשורה העליונה).
3.הזזת כל החיילים משבצת אחת ימינה (החיילים שבטור הימני עוברים לטור השמאלי).
4.הזזת כל החיילים משבצת אחת שמאלה (החיילים שבטור השמאלי עוברים טור הימני).
5.ערבוב כל החיילים על גבי הלוח בסידור אקראי (אף חייל לאחר הערבוב לא יופיע על גבי משבצת בצבע שלו).
לאחר ביצוע תור של כל שחקן נבצע בדיקה שבה נבדוק האם ישנם חיילים שנמצאים על גבי משבצות באותו הצבע כמו צבעי החיילים, ואם כן שחקנים אלה יצאו מהמשחק ולאחר מכן משבצות אלו יתפנו וייצבעו בצבע לבן.
המשחק ימשיך לרוץ על פי תוכנית זו עד שיגיע מצב שבו לאחד השחקנים נותרו פחות חיילים ממחצית המשבצות בכל שורה\עמודה.
במידה וישנו מצב שבו 2 שחקנים (או יותר) עומדים בתנאי ניצחון זה אזי השחקן המנצח יהיה השחקן בעל מספר החיילים הקטן ביותר ,אך במידה ולשניהם(או יותר) יש אותם מספר חיילים אזי ישנו תיקו והם יוכרזו כמנצחים ביחד.

פירוט האתגרים השונים:
אתגר ראשון - לאחר שפונקציית randi שמפזרת את המספרים מ1 עד מספר השחקנים (באופן אקראי כך שלא יהיה ניתן לדעת כמה פעמים יש לנו מספר מסוים) שנבחרו על גבי הלוח שנקבע .
עלה בפנינו אתגר שבו אנו נצטרך לגרום למטריצה שיצרנו למצב שהיא תחליף במקומות הלא רצויים\ מיותרים (כלומר המספרים שהמטריצה יצרה לנו יותר מהמספר המקסימלי שנקבע עבור חיילים\קוביות צבועות) את המספרים הללו במספרים שחסר לנו מהם בשביל להגיע למספר המקסימלי של החיילים\קוביות עבור הלוח שנקבע.
כך שכל מספר בלוח שמציין את החיילים\קוביות צבועות לכל שחקן יחולק באופן שווה , וזה בעצם יתאר לנו את כמות החיילים\קוביות צבועות שיש לכל שחקן.

אתגר שני - כאשר אנו מחברים את הלוחות של החיילים שמפוזרים על גבי הלוח, בלוח השני שבו צבועים צבעי הקוביות בצבעי החיילים שנקבעו נדאג שאין מצב שבו חיילים בעלי צבע מסוים נמצאים על קוביות שצבועות באותו הצבע כמו שלהם כי אחרת יאלצו להימחק מן המשחק עוד לפני שהמשחק התחיל ומצב זה לא ייתכן.

אתגר שלישי - ליצור מהלך שיאפשר לשחקן בתורו לעשות ערבוב של כל החיילים על גבי לוח המשחק אבל בידיעה שאין אפשרות לכך שהחיילים בצבע של השחקן ששיחק יפלו על גבי קוביות באותו הצבע.

אתגר רביעי - ביצוע מהלך של גריעת החיילים לאחר שהשחקן שיחק וגרם לכך שיהיו חיילים על גבי משבצת בצבע זהה ובנוסף לפנות את צבע הקובייה ולהפוך אותה ללבנה .

אתגר חמישי - לבדוק במצב שבו ישנם 2 שחקנים או יותר שמקיימים את תנאי הניצחון למי יש פחות שחקנים על גבי הלוח והוא יהיה המנצח הסופי או שיהיה תיקו בין השחקנים בעלי מספר השחקנים הנמוך ביותר והם יוכרזו כמנצחים.

אסטרטגיות לפתרון האתגרים:

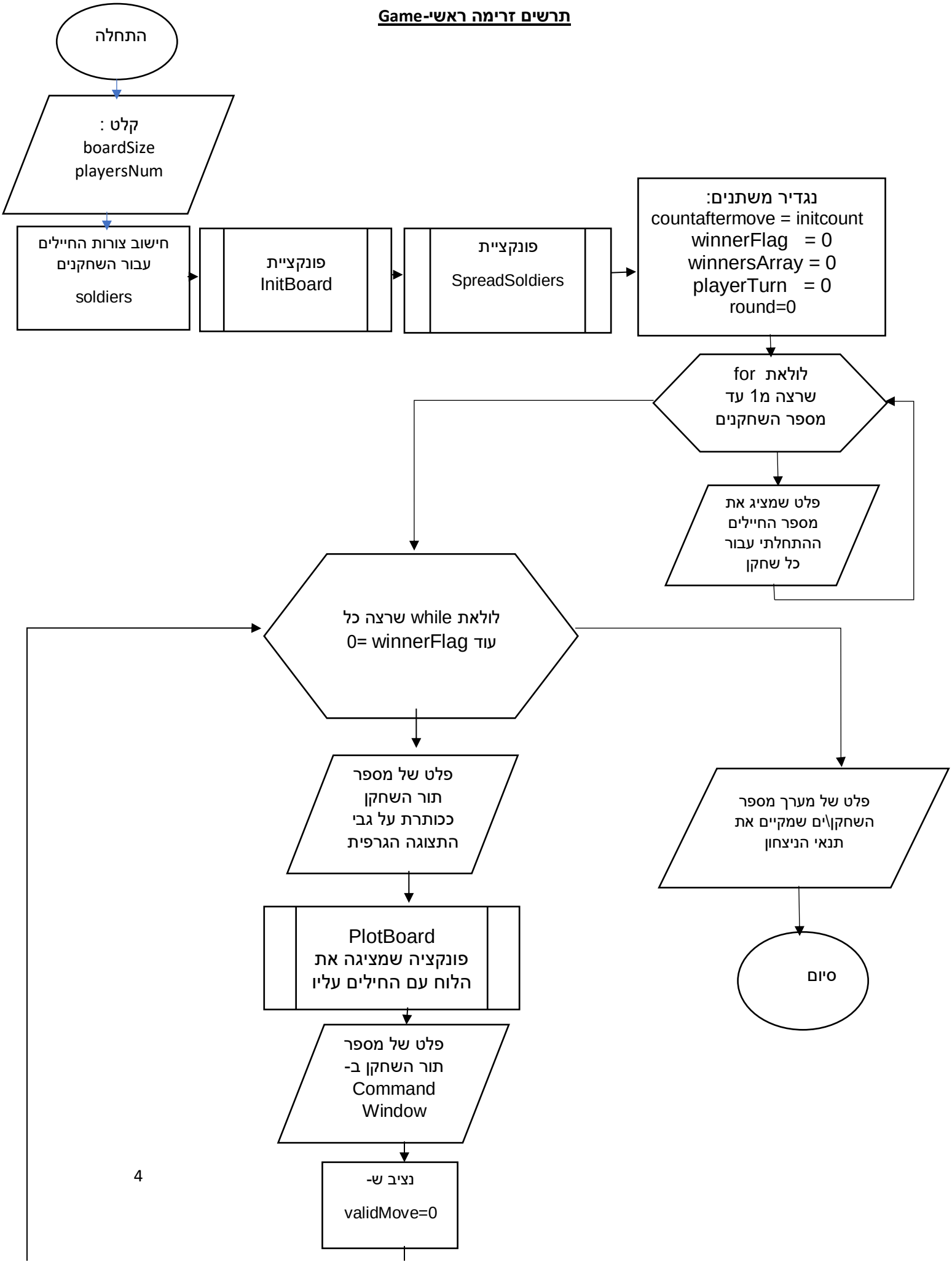
אתגר ראשון - כדי לפתור אתגר זה עלינו לבדוק כמה פעמים מופיע מספר מסוים על גבי המטריצה(שקיבלנו randin שזהו לוח המשחק) . עלינו לדאוג לכך שיהיה מספר חיילים שווה עבור כל השחקנים ומספר קוביות צבועות שווה לכל שחקן. אופן כיוון הפתרון שלנו עבור אתגר זה ייעשה באמצעות בדיקה של איבר איבר בלוח (באמצעות לולאה שרצה על מספר העמודות ולולאה שרצה על מספר השורות) , וכאשר נגיע למספר החיילים המקסימלי שחושב עבור כל שחקן נבדוק שקיימים מספר חיילים וקוביות צבועות זהות נעבור לשחקן הבא וכך הלאה עד כדי הגעה למצב שלכולם יש מספר שווה של חיילים וקוביות צבועות. ובמידה ונשארו לנו קוביות מיותרות נסמן אותם במספר 0 שזה יהיה עבורנו קוביות ריקות שייצבעו בצבע לבן.

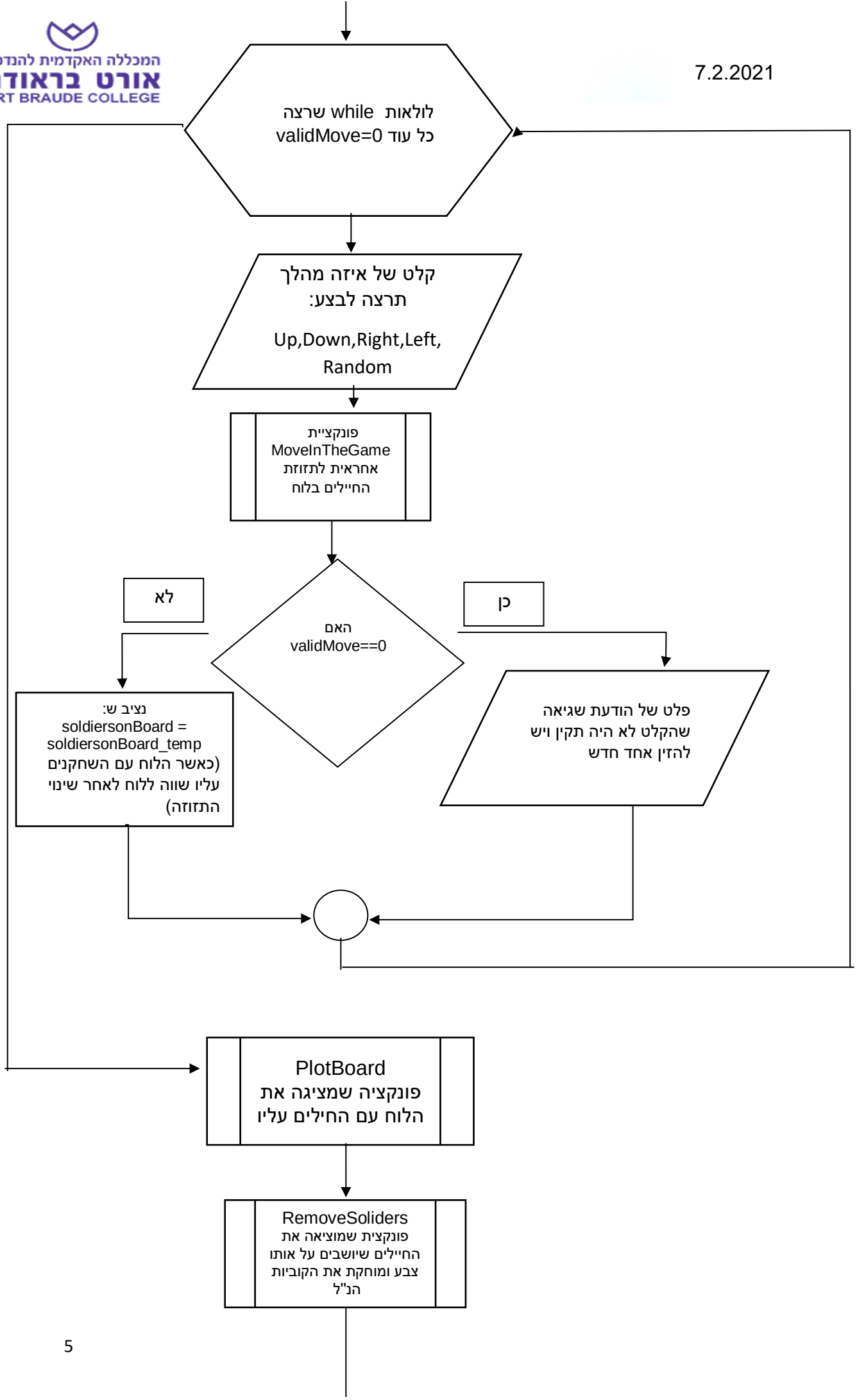
אתגר שני + אתגר שלישי - נשים לב שכאשר נפתור את האתגר השני זה יוכל להוות פתרון גם עבור האתגר השלישי (כאשר נגדיר את האפשרות של ערבוב אקראי של החיילים על גבי הלוח כפונקציה שתפתור לנו את האתגר השני). נעשה בדיקה של איבר איבר על גבי הלוח (בהרצה על גבי השורות והעמודות) שאין חייל שיושב על קובייה באותו צבע כמו שלו, ננסה לבצע זאת באמצעות כך שנציב מספרים באופן רנדומלי על לוח חדש ואם המספר שהוצב שונה מאפס וממספר של הלוח המקורי במקום(i,j) אזי נציב אותו בלוח החדש באותו מקום ואם זה יהיה שווה לאפס נציב במקום זה על הלוח את 0 וכך נדאג למלא את הלוח. בנוסף לכך כדי שנדאג שלכל שחקן יהיה מספר שווה של חיילים נדרוש בנוסף שהמערך שסופר יהיה יותר קטן מהמערך שנקבל מהפונקציה של הלוח המקורי של כמה חיילים מותר לכל שחקן.

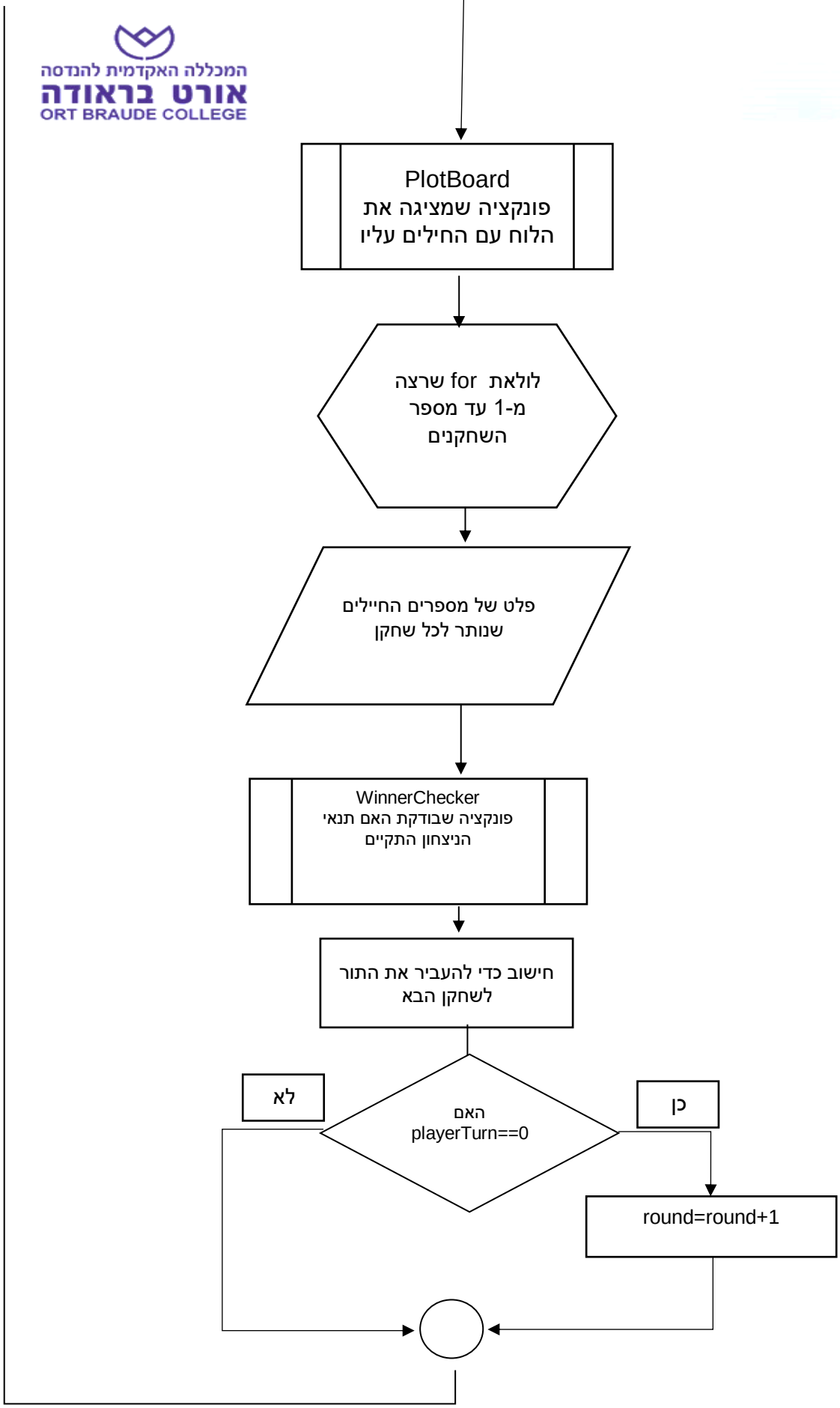
אתגר רביעי- בדיקה לאחר ביצוע של מהלך האם ישנם חיילים שנמצאים על גבי משבצות מאותה הצבע , בדיקה זו תיעשה באמצעות לולאות שיעברו על גבי השורות ועל גבי העמודות ויבדקו היכן המספרים הללו יהיו זהים על גבי הלוח המקורי אל מול הלוח לאחר התזוזות. בדיקה זאת תתבצע ותעבור איבר איבר על גבי לוח(בהרצה על גבי השורות והעמודות) והיכן שישנה התאמה נמחק את החיילים הנ"ל ונפנה את הקובייה ונצבע אותה בצבע לבן.

אתגר חמישי- בדיקה לאחר קיום תנאי הניצחון כמה שחקנים יש לכל שחקן באמצעות פונקציית sum וכך נבדוק למי יש את מספר השחקנים הקטן ביותר והשחקן בעל מספר זה יוכרז כמנצח, בנוסף במידה וישנם 2 שחקנים (או יותר) שיש להם מספר זהה של שחקנים לאחר הבדיקה הם יוכרזו כמנצחים ביחד(תיקו).

תרשים זרימה ראשי-Game




לולאות while שרצה
כל עוד validMove=0
קלט של איזה מהלך
תרצה לבצע:
Up,Down,Right,Left,
Random
פונקציית
MoveInTheGame
אחראית לתזוזת
החיילים בלוח
לא
כן
האם
validMove==0
נציב ש:
soldiersonBoard =
soldiersonBoard_temp
(כאשר הלוח עם השחקנים
עליו שווה ללוח לאחר שינוי
התזוזה)
פלט של הודעת שגיאה
שהקלט לא היה תקין ויש
להזין אחד חדש
PlotBoard
פונקציה שמציגה את
הלוח עם החילים עליו
RemoveSoliders
פונקצית שמוציאה את
החיילים שיושבים על אותו
צבע ומוחקת את הקוביות
הנ"ל


PlotBoard
פונקציה שמציגה את
הלוח עם החילים עליו
לולאת for שרצה
מ-1 עד מספר
השחקנים
פלט של מספרים החיילים
שנותר לכל שחקן
WinnerChecker
פונקציה שבודקת האם תנאי
הניצחון התקיים
חישוב כדי להעביר את התור
לשחקן הבא
האם
playerTurn==0
לא
כן
round=round+1

אופן פעולה של התוכנית הראשית-Game

מגדירים בתחילת המשחק כמה שחקנים יהיו (playerNum) ומה יהיה גודל הלוח(boardSize),לאחר מכן נגדיר את צורות החיילים עבור כל שחקן.

לאחר מכן פונקציית InitBoard תהיה אחראית על בדיקת מהו מספר המקסימלי של חיילים שמגיע לכל שחקן ,ובנוסף לחלק את כל הלוח באופן שווה לקוביות צבועות לפי מספר החיילים שנקבע לכל שחקן כדי שיהיה לנו התאמה בין הקוביות הצבועות לבין מספר החיילים.

לאחר מכן נזמן את פונקציית SpreadSoliders שתפקידה יהיה למקם את החיילים באופן אקראי על הלוח כך שאף חייל לא ייפול על הצבע שלו מאחר ואם הוא ייפול על הצבע שלו נתקל בבעיה שהוא יימחק עוד לפני תחילת המשחק.

הגדרת אתחול תנאי המשחק : countaftermove יהיה אחראי לספירת הקוביות הצבועות שיש בלוח, WinnerFlag אחראי לכך שנדע אם יש שחקן שניצח (כל עוד שונה מ1 המשחק ממשיך להתקיים) , winnersArray אחראי למסור את מספר השחקן\נים שניצחו , playerTurn אחראי להציג את תור השחקן , round אחראי לייצג את מספר הסבב.

נשתמש בלולאת for שרצה על כל השחקנים כדי להציג את הכמות חיילים ההתחלתית שיש לכל שחקן.

לאחר מכן נשתמש בלולאת while אשר תפקידה לבדוק האם תנאי הניצחון מתקיים (כאשר לאחד מהשחקנים נותר פחות חיילים ממחצית שורה/עמודה). בתוך לולאה זו נציג לשחקן את מספר הסבב ואת מספר התור של כל שחקן בתצוגה הגרפית (בעזרת פונקציית PlotBoard).

לאחר מכן נציג לשחקן את אופציית התזוזה במשחק ונבדוק את תקינות ההזנת התזוזה באמצעות לולאת while שכאשר המשתנה validMove שונה מ1 אזי התזוזה תקינה.

לאחר שהשחקן מבצע את התזוזה שלו נציג לשחקן את הפעולה שלו איך היא בוצעה על הלוח בתצוגה הגרפית למשך 3 שניות.

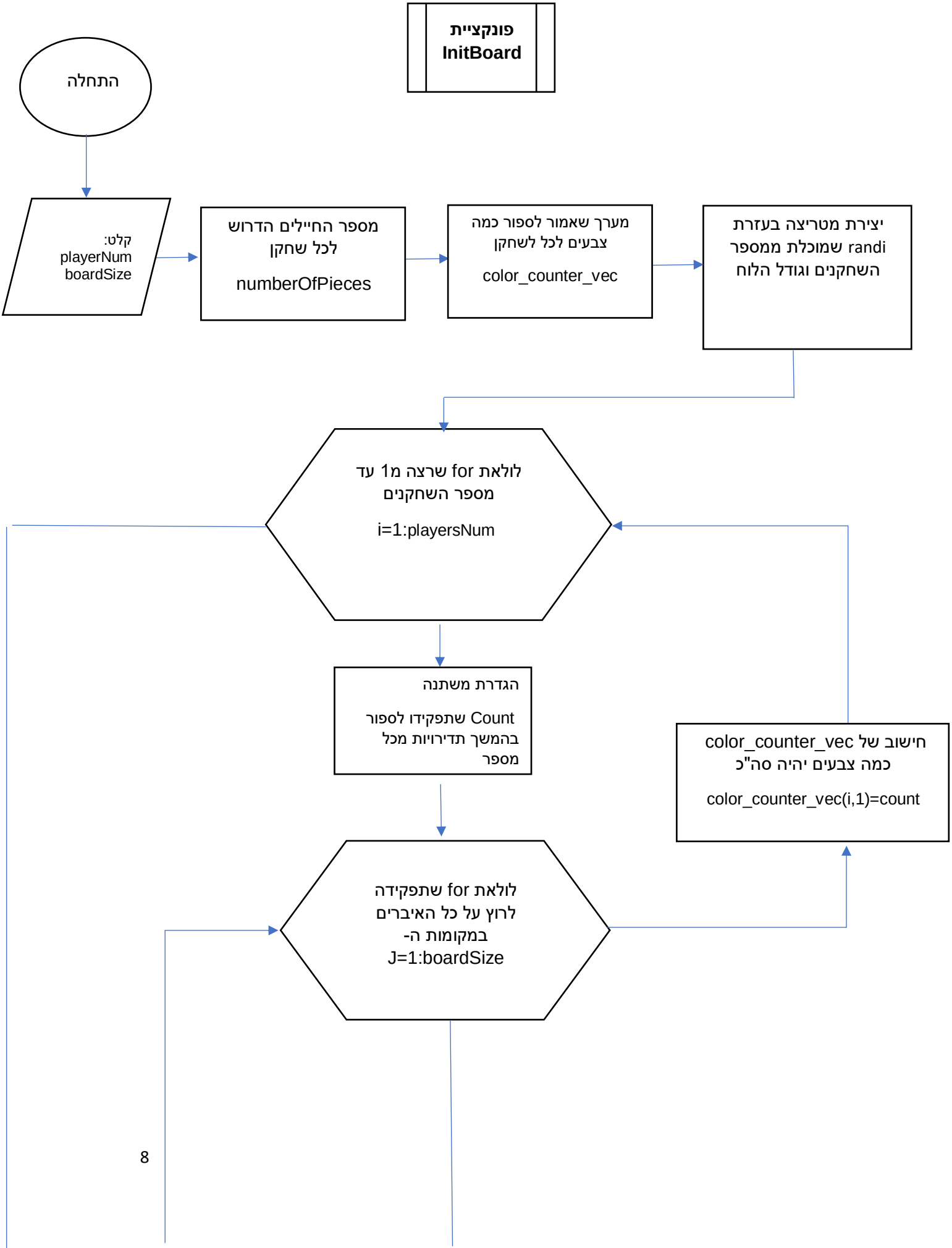
בשלב הבא נשתמש בפונקציית RemoveSoldiers שתבדוק איזה שחקנים צריכים להימחק מהלוח, ואז נציג באמצעות התצוגה הגרפית את החיילים שיימחקו בסימון של עיגול לבן ואז נציג את הלוח לאחר מחיקתם.

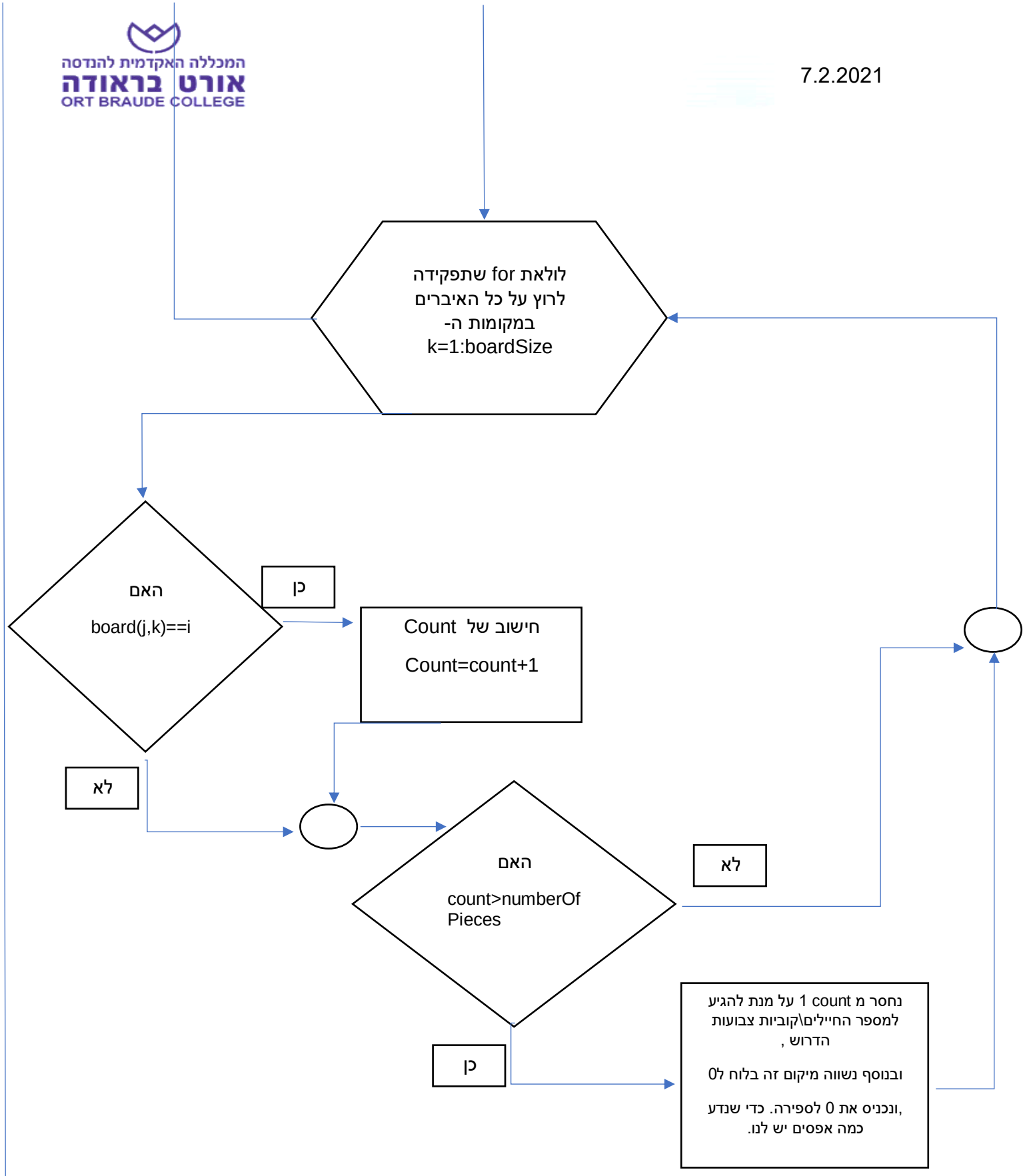
נציג לשחקנים כל חיילים נשארו לכל שחקן על ידי תצוגה בcommand window.

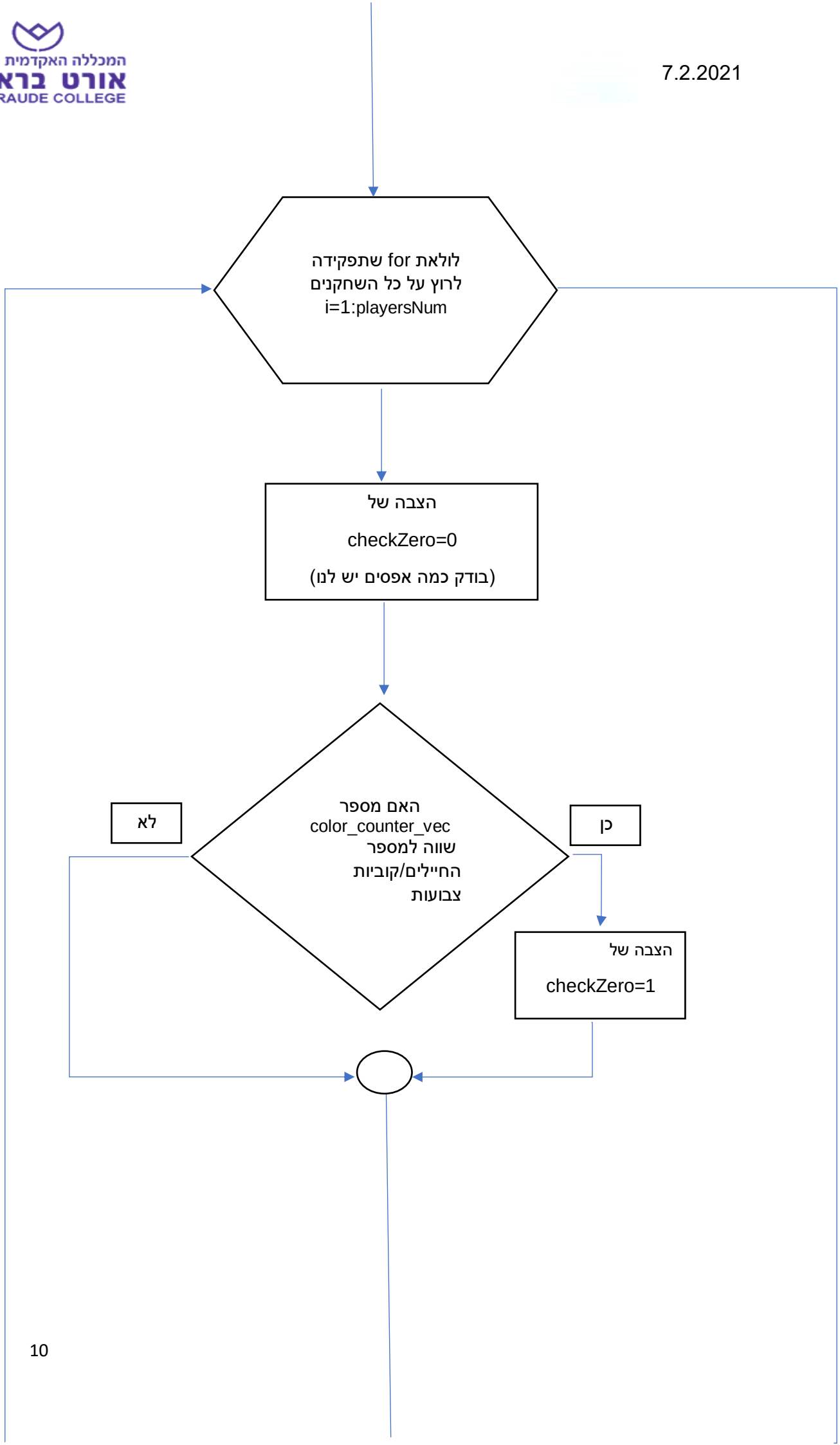
בשלב הבא נבדוק האם תנאי הניצחון התקיים עבור אחד השחקנים, ועבור זה נשתמש בפונקציית WinnerChecker כדי לבדוק זאת, במידה ותנאי הניצחון לא התקיים נבצע את חישוב של מעבר לשחקן הבא בכך שנשתמש בrem על ידי חילוק playerTurn במספר השחקנים .

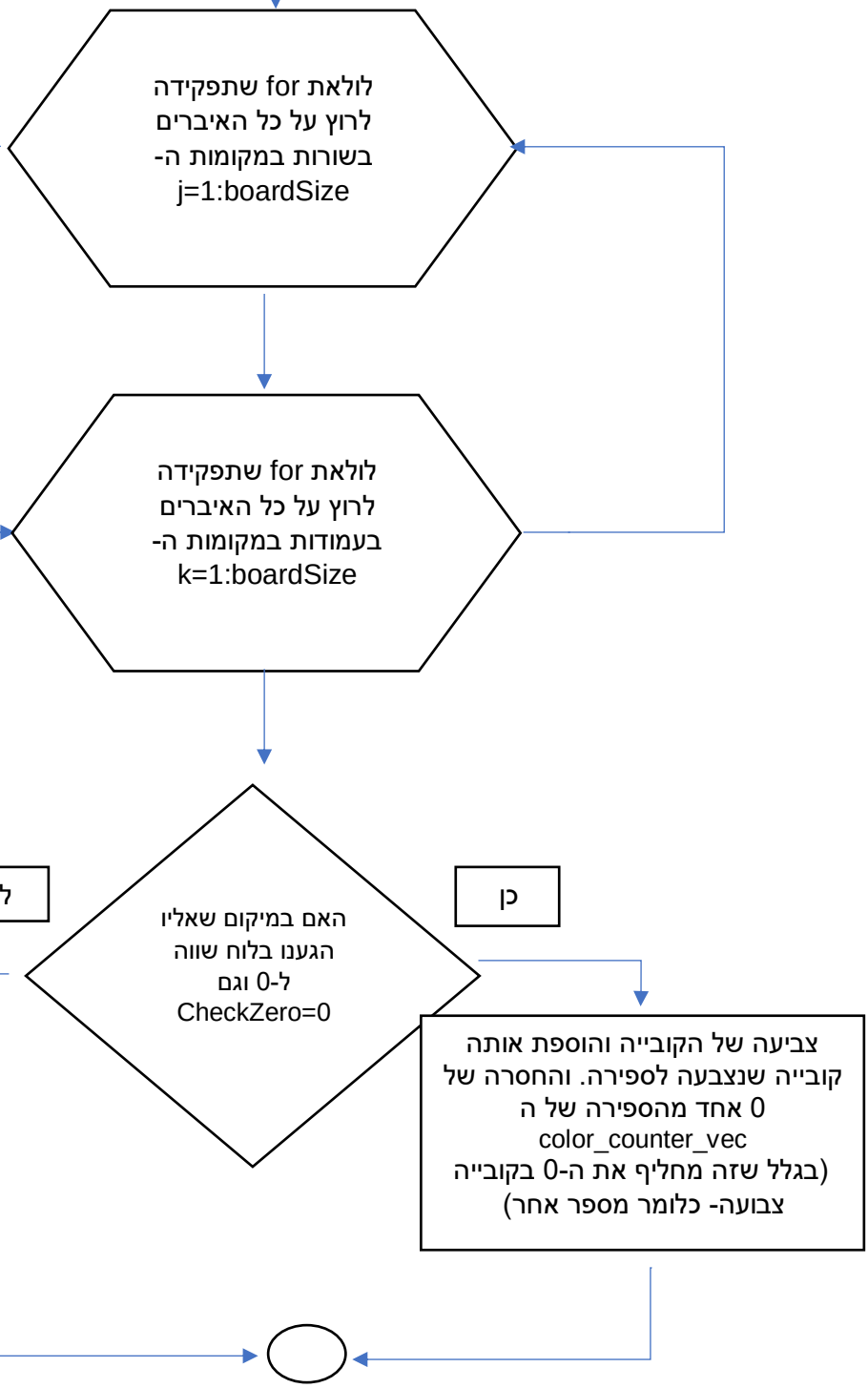
לאחר מכן נכנס לתנאי שיבדוק האם כל השחקנים שיחקו ויש לעבור לסיבוב הבא.

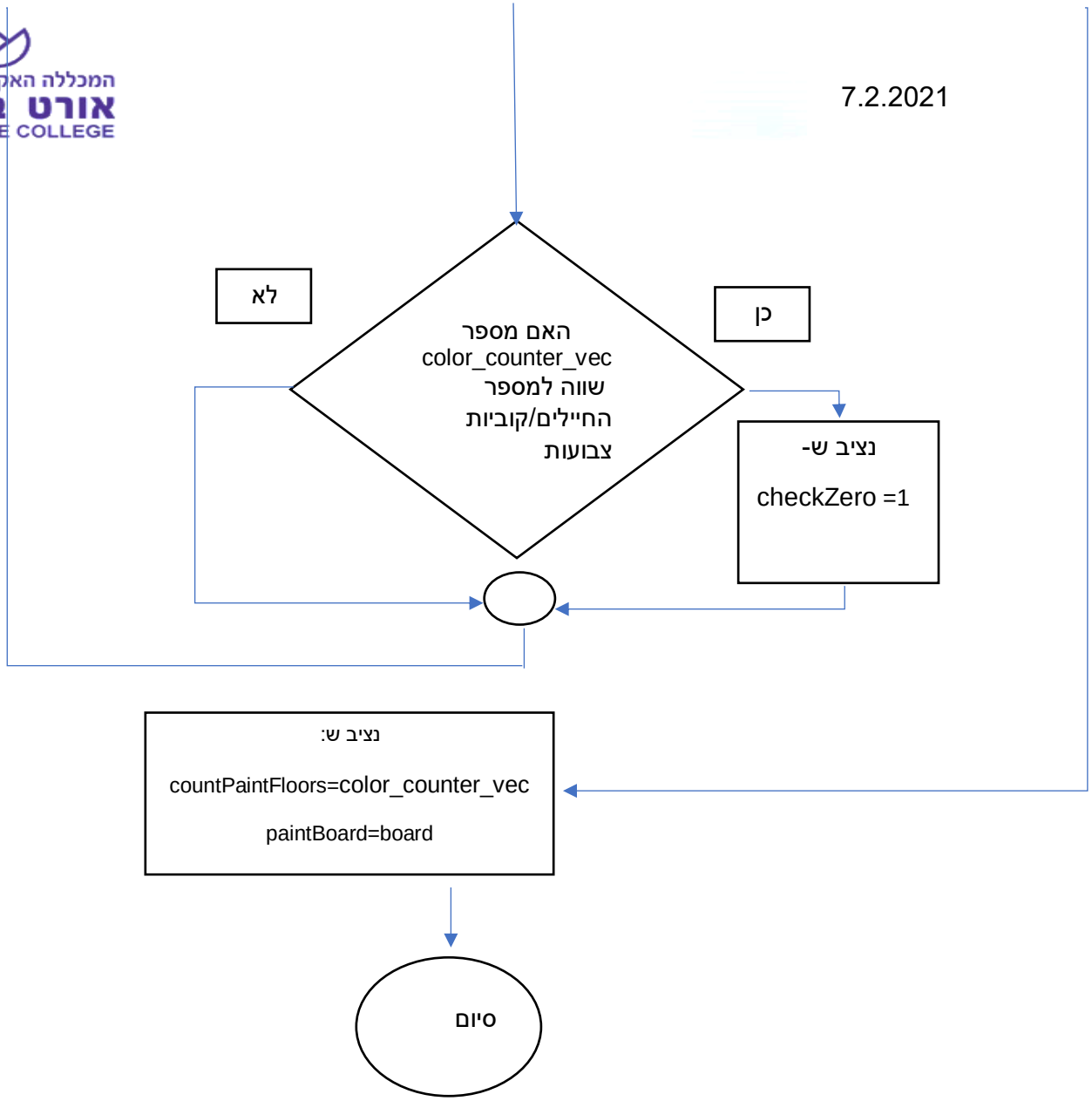
לבסוף כאשר תנאי הניצחון מתקיים נדפיס הודעות ניצחון ב command window עם מספר השחקן \ נים שניצחו.


פונקציית
InitBoard
התחלה
קלט:
playerNum
boardSize
מספר החיילים הדרוש
לכל שחקן
numberOfPieces
מערך שאמור לספור כמה
צבעים לכל לשחקן
color_counter_vec
יצירת מטריצה בעזרת
randi שמוכלת ממספר
השחקנים וגודל הלוח
לולאת for שרצה מ1 עד
מספר השחקנים
i=1:playersNum
הגדרת משתנה
Count שתפקידו לספור
בהמשך תדירויות מכל
מספר
חישוב של color_counter_vec
כמה צבעים יהיה סה"כ
color_counter_vec(i,1)=count
לולאת for שתפקידה
לרוץ על כל האיברים
במקומות ה-
J=1:boardSize


לולאת for שתפקידה
לרוץ על כל האיברים
במקומות ה-
k=1:boardSize
האם
board(j,k)==i
כן
חישוב של Count
Count=count+1
לא
האם
count>numberOf
Pieces
לא
כן
נחסר מ count 1 על מנת להגיע
למספר החיילים\קוביות צבועות
הדרוש ,
ובנוסף נשווה מיקום זה בלוח ל0
,ונכניס את 0 לספירה. כדי שנדע
כמה אפסים יש לנו.


לולאת for שתפקידה
לרוץ על כל השחקנים
i=1:playersNum
הצבה של
checkZero=0
(בודק כמה אפסים יש לנו)
האם מספר
color_counter_vec
שווה למספר
החיילים/קוביות
צבועות
לא
כן
הצבה של
checkZero=1


לולאת for שתפקידה
לרוץ על כל האיברים
בשורות במקומות ה-
j=1:boardSize
לולאת for שתפקידה
לרוץ על כל האיברים
בעמודות במקומות ה-
k=1:boardSize
האם במיקום שאליו
הגענו בלוח שווה
ל-0 וגם
CheckZero=0
לא
כן
צביעה של הקובייה והוספת אותה
קובייה שנצבעה לספירה. והחסרה של
0 אחד מהספירה של ה
color_counter_vec
(בגלל שזה מחליף את ה-0 בקובייה
צבועה- כלומר מספר אחר)


לא
כן
האם מספר
color_counter_vec
שווה למספר
החיילים/קוביות
צבועות
נציב ש-
checkZero =1
נציב ש:
countPaintFloors=color_counter_vec
paintBoard=board
סיום

פונקציית InitBoard

אופן הפעולה של פונקציית InitBoard היא תקבל את מספר השחקנים ואת גודל הלוח, לאחר מכן על מנת למצוא את המספר החיילים כמספר הגדול ביותר שמאפשר סידור אקראי של כל החיילים מבלי להניח אף חייל על משבצת בצבע שלו. נקרא לו בשם numberOfPieces והוא יקבע כמה חיילים יהיה לכל שחקן בנוסף לכך נשתמש ב color_counter_vec שתפקידו יהיה לספור כמה קוביות צבעות יהיה לכל שחקן.

חישוב numberOfPieces יתבצע על ידי הנוסחה – fix(boardSize^2/playersNum)

נייצר מטריצת randi אשר תכלול בתוכה את המספרים מ-1 עד מספר השחקנים ומספר השורות והעמודות יהיה אותו מספר כמו המספר שהוזן כגודל הלוח לדוגמא (לוח בגודל 7 אזי המטריצה תהיה 7X7) שהוא בעצם תהיה הלוח הראשי שלנו, שיקרא(InitBoard).

לאחר מכן כדי שנדאג שלכל שחקן יהיה את אותו מספר כמות של קוביות צבועות נשתמש בלולאת for שתפקידה יהיה לרוץ מ-1 עד מספר השחקנים שנקרא לו (playersNum) שבעצם תפקיד לולאה זו יהיה לספור כמה קוביות צבועות יש לנו ואם יש עודף ממה שצריך אזי נחליף את המיקומים האלה במספר 0.

בתוך לולאה זו נצטרך להשתמש בלולאה שתרוץ על כל השורות (j=1:boardSize) ,ובתוכה לולאה נוספת שתרוץ על כל העמודות (k=1:boardSize) שהם יתחילו מ-1 עד לגודל הלוח.

לאחר מכן נבדוק אם מתקיים התנאי שבו אם המיקום בתוך המטריצה((board(j,k) יהיה שווה למספר של אותו שחקן שאנחנו רצים אליו מתחילת הלולאה אזי נוסיף לו 1 כלומר (count=count+1) וכך אנחנו בעצם סופרים כמה קוביות יש לכל שחקן.

בנוסף לכך במקרה ובו מספר הקוביות יהיה יותר גדול ממספר החיילים שנקבע מלכתחילה אזי נצטרך לחסר אחד בספירה שלנו כדי להחזיר את זה למצב התקין ואז באותו מקום אנחנו נדרוש שבמקום המספר שמופיע במטריצה נחליף אותו באפס ונכניס לספירה שלנו את מספר האפסים שיש לנו.

לאחר שתסתיים פעולה זו עבור שחקן ראשון נרצה לאחסן את המידע הזה במשתנה ב color_counter_vec(i,1)=count כדי לדעת כמה יש לנו מכל צבע (כלומר כמה יש לנו מכל מספר), וכך לולאה ראשית זו תרוץ מ-1 עד playersNum ותבדוק את כל השחקנים.

במהלך פונקציה זו נוצרה לנו בעיה כרגע שעדיין לא הצלחנו להגיע למספר שווה לכל השחקנים של קוביות צבועות נצטרך להפוך את כל האפסים שנוצרו לנו למספרים אשר יש פחות ממספר הקוביות הצבועות הנדרש.

נעזר בלולאת for שתפקידה יהיה לחפש על אותם אפסים לאחר שנגמרה הלולאה הראשית והיא תרוץ מ- 1:playersNum ,נגדיר משתנה בשם checkZero=0 ואם המספר של color_counter_vec(i,1)=numberOfPieces אזי checkZero=1 וכך אנחנו נמנע מעצמנו לחזור על אותה בדיקה פעם נוספת.

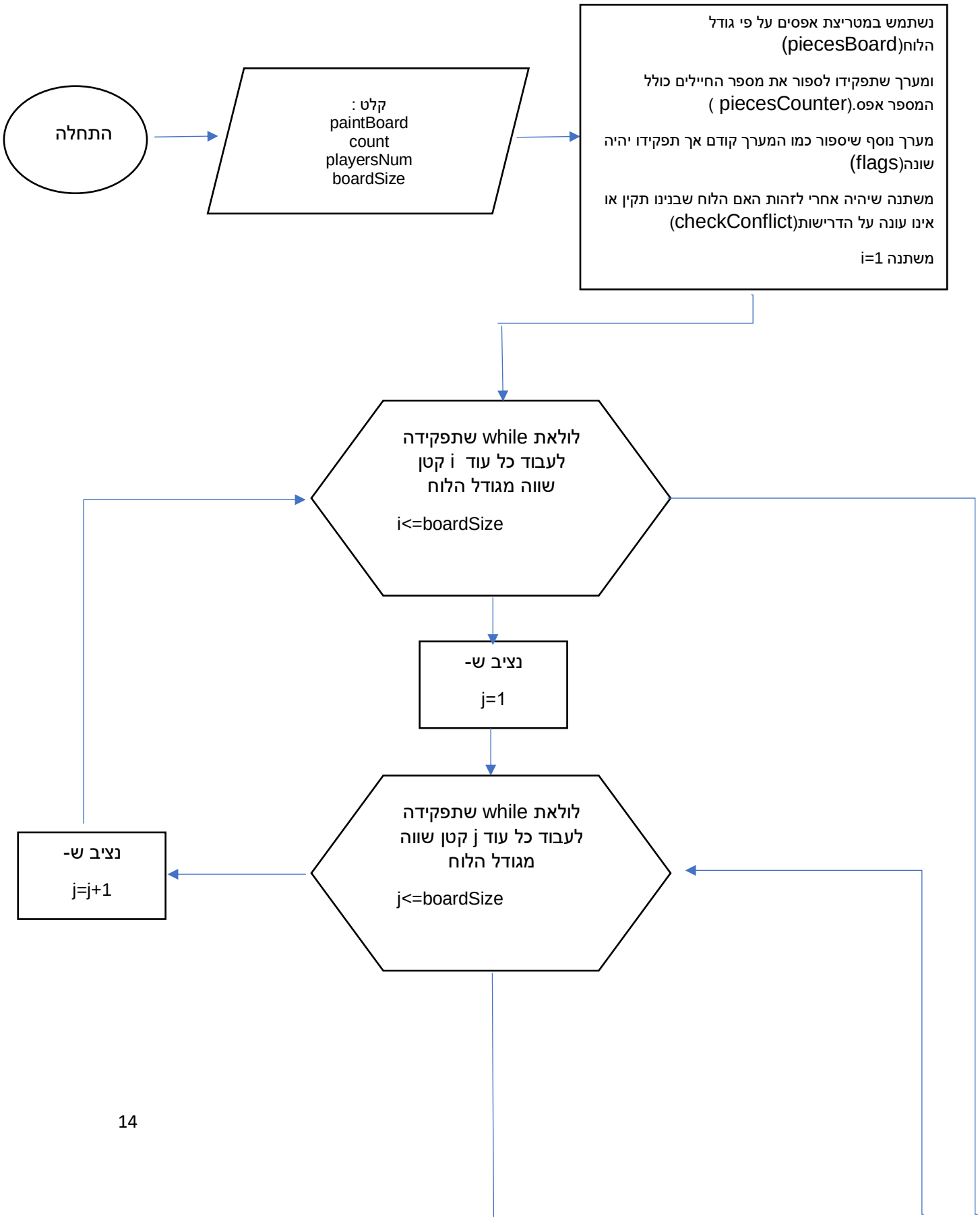
לאחר מכן נשתמש בלולאת for עבור השורות שתפקידה יהיה לרוץ מ-j=1:boardSize ובתוכה תהיה עוד לולאה שתרוץ על כל העמודות k=1:boardSize ואם במיקום (j,k)על הלוח אנחנו נהיה שווים לאפס וגם checkZero=0 ,שזה בעצם אומר שהוא לא צבוע המיקום הזה אזי נצבע את המיקום הזה במספר של השחקן שאליו אנחנו עושים את הבדיקה ולאחר מכן נוסיף את הקובייה החדשה שצבענו color_counter_vec(i,1)=color_counter_vec(i,1)+1 .

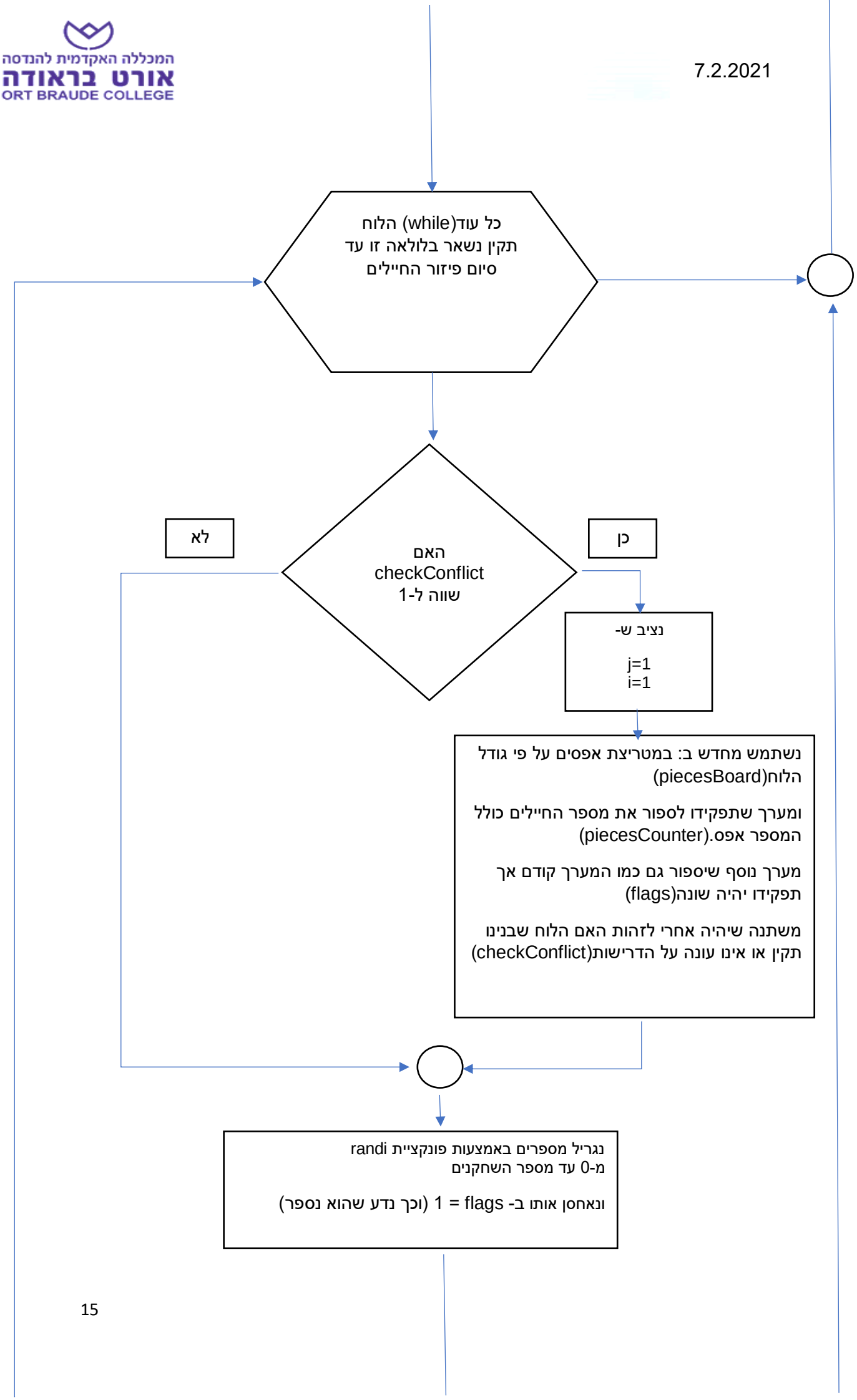
בדרך זו נכניס לספירה הכוללת ולבסוף נחסיר את אותו אפס שצבענו מאותו סכום אפסים שיש לנו. ואם יש לנו מספר צבעים ששווה למספר החיילים אזי נכריז על זה שזה מצב תקין ונמשיך הלאה (checkZero=1) ,כל הפעולות האלה יתבצעו על לולאת ה for המשנית שתרוץ על כל השחקנים.

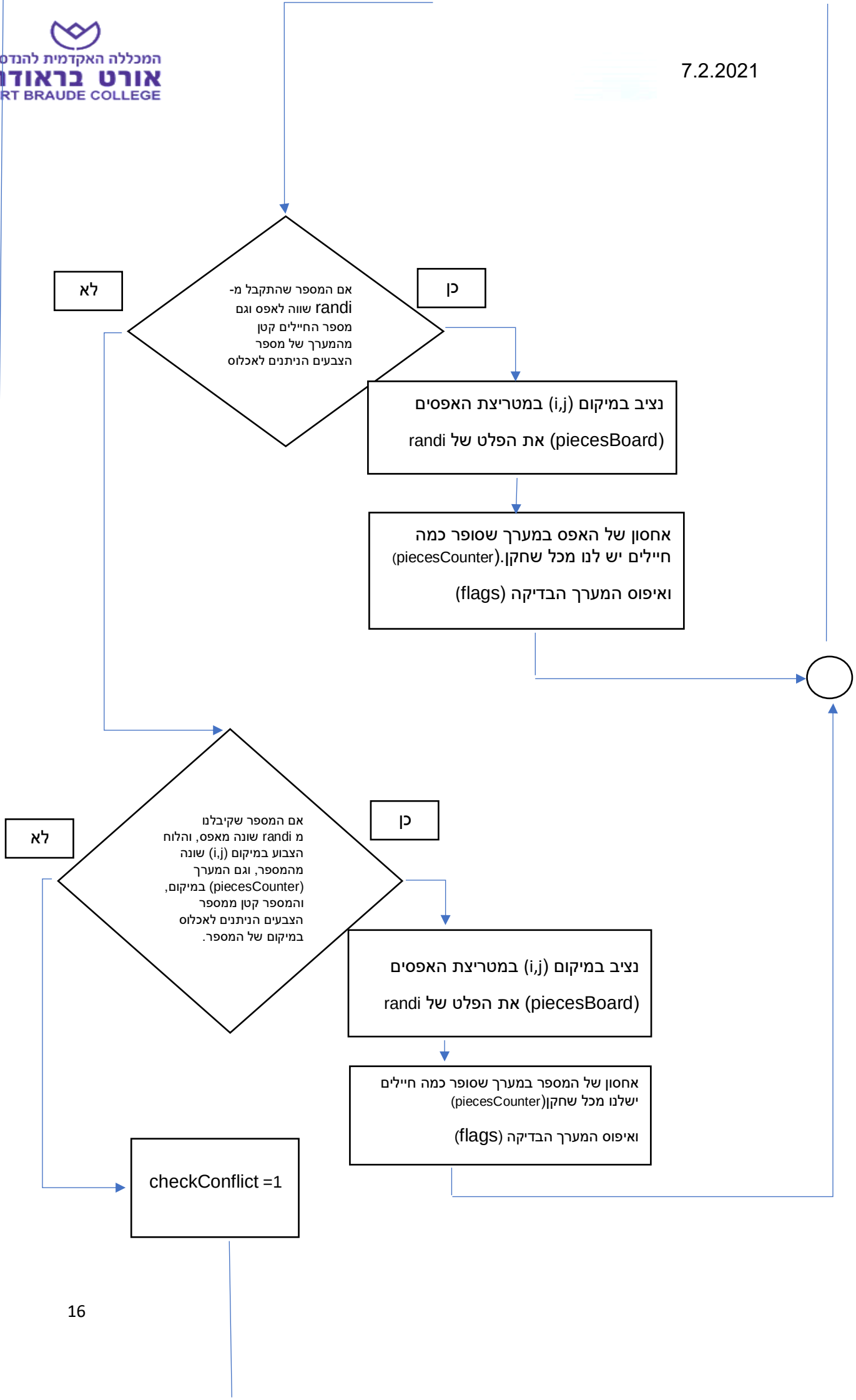
לאחר שזה יגמר נדרוש לקבל בחזרה את מספר הקוביות הצבעות ואת הלוח הצבוע.

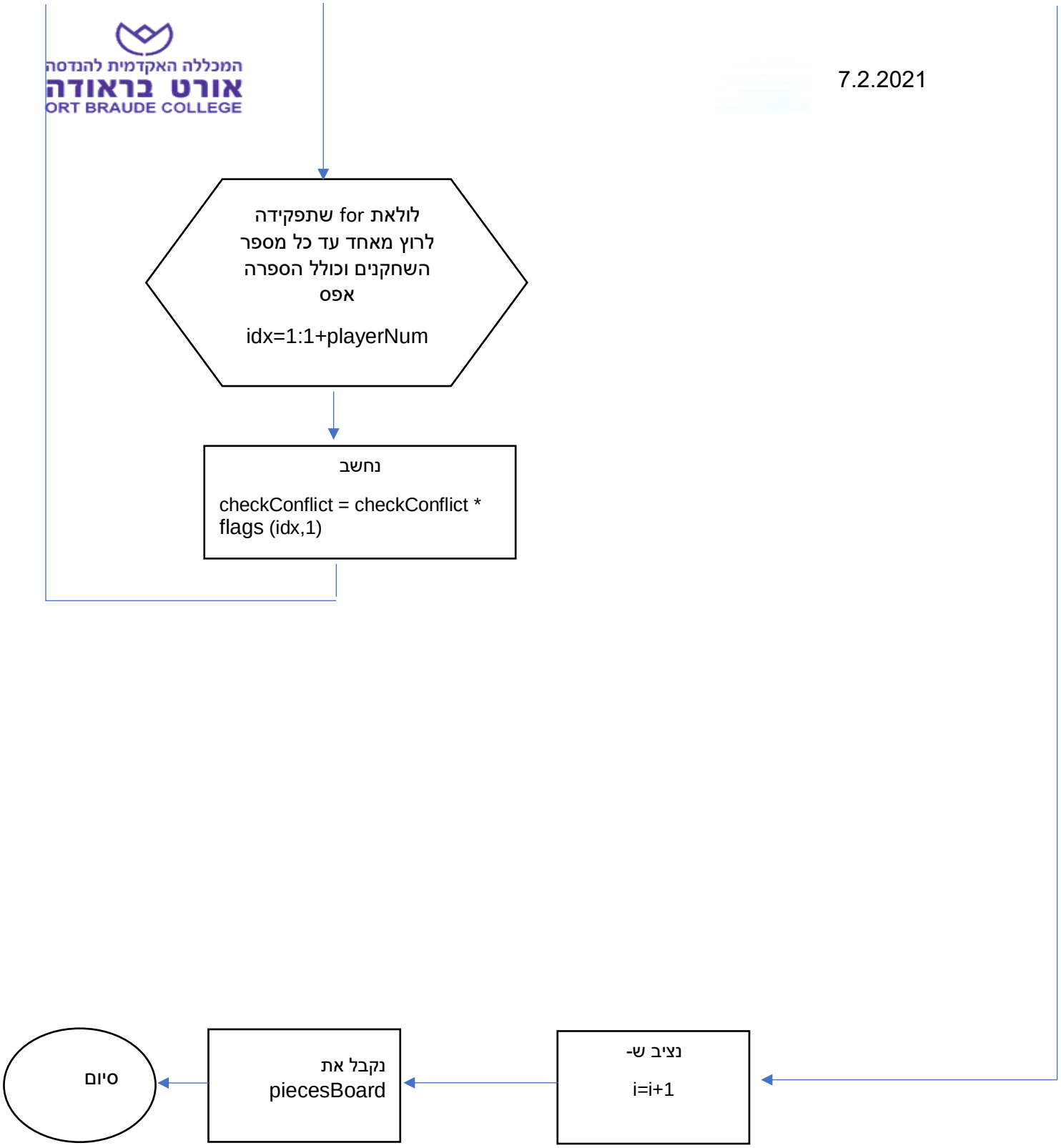
פונקציית

SpreadSoldiers




כל עוד(while) הלוח
תקין נשאר בלולאה זו עד
סיום פיזור החיילים
האם
checkConflict
שווה ל-1
לא
כן
נציב ש-
j=1
i=1
נשתמש מחדש ב: במטריצת אפסים על פי גודל הלוח(piecesBoard)
ומערך שתפקידו לספור את מספר החיילים כולל המספר אפס.(piecesCounter)
מערך נוסף שיספור גם כמו המערך קודם אך תפקידו יהיה שונה(flags)
משתנה שיהיה אחרי לזהות האם הלוח שבנינו תקין או אינו עונה על הדרישות(checkConflict)
נגריל מספרים באמצעות פונקציית randi
מ-0 עד מספר השחקנים
ונאחסן אותו ב- flags = 1 (וכך נדע שהוא נספר)


אם המספר שהתקבל מ-randi שווה לאפס וגם מספר החיילים קטן מהמערך של מספר הצבעים הניתנים לאכלוס
כן
לא
נציב במיקום (i,j) במטריצת האפסים (piecesBoard) את הפלט של randi
אחסון של האפס במערך שסופר כמה חיילים יש לנו מכל שחקן.(piecesCounter) ואיפוס המערך הבדיקה (flags)
אם המספר שקיבלנו מ randi שונה מאפס, והלוח הצבוע במיקום (i,j) שונה מהמספר, וגם המערך (piecesCounter) במיקום, והמספר קטן ממספר הצבעים הניתנים לאכלוס במיקום של המספר.
כן
לא
נציב במיקום (i,j) במטריצת האפסים (piecesBoard) את הפלט של randi
אחסון של המספר במערך שסופר כמה חיילים ישלנו מכל שחקן(piecesCounter) ואיפוס המערך הבדיקה (flags)
checkConflict =1


לולאת for שתפקידה
לרוץ מאחד עד כל מספר
השחקנים וכולל הספרה
אפס
idx=1:1+playerNum
נחשב
checkConflict = checkConflict *
flags (idx,1)
נציב ש-
i=i+1
נקבל את
piecesBoard
סיום

פונקציית SpreadSoliders

פונקציה זו תהיה אחראית על פיזור של החיילים על הלוח שנבנה באופן רנדומלי מבלי שאותו חייל יהיה ממוקם על אותו צבע שלו. פונקציה זו תקלוט את הערכים של הלוח הצבוע שנבנה בפונקציית InitBoard ובנוסף את המערך שסופר את כמות הקוביות הצבועות שניתן לאכלס אליהם חיילים, וגם מספר השחקנים ולבסוף את גודל הלוח.

נגדיר : מטריצת אפסים על פי גודל הלוח שתיקרא – piecesBoard מערך שתקפידו לספור את מספר החיילים (כולל המספר אפס שמקומו במערך יהיה האחרון: במיקום מספר השחקנים+1) שיקרא –piecesCounter , ומשתנה בשם flags שמטרתו לבדוק אם ישלנו לולאה אינסופית, ומשתנה עזר שישתנה בהתאם לflags ויקרא-checkConflict .

אם כל איברי flags הם אחדים אז checkConflict שווה 1 אחרת שווה 0

נתחיל את הספירה מ i=1 ונכנס ללולאת while שכל עוד היא יותר קטנה מגודל הלוח היא תעבוד , נעשה זאת על השורות וגם על העמודות ולאחר מכן תהיה לולאת while שתפקידה יהיה לבדוק מספרים רנדומליים מהמספרים האפשריים כדי להציב אותם בנקודה (i,j) כך בלולאה זו נגריל מספר בעזרת פונקציית randi.

לאחר מכן נגדיר flags מוודא שכל מספר שנעשה עליו את הבדיקה אם הוא מתאים בלוח יסומן כאחד וכך נדע שהמספר הזה נוסה. ובנוסף מטרתו לבדוק אם יש לנו לולאה אינסופית כך שבהתחלה הוא מאותחל לאפסים כאשר כל צבע של חייל שננסה נסמן ב flags את מיקומו כ-1 לדוגמא: אם ננסה לשים חייל בצבע 2 אז flags במקום ה-2 יהיה 1.

אם המספר שהוגרל שווה ל0 וגם מספר החיילים קטן ממספר המערך שמכיל בתוכו כמה קוביות צבועות מותר שיהיה לכל שחקן אזי באותו מיקום במטריצה נוסיף את המספר שהוגרל ונספור את 0 במערך (piecesCounter) ,ונאפס את ה flags כי הצלחנו להציב מספר בתוך המטריצה וחוזרים ללולאה של העמודות בשביל לעבור לעמודה הבאה.

אם המספר שהוגרל שונה מ-0 והמיקום על הלוח שונה מהמספר שהוגרל וגם מספר השימושים בו (הצבתו בלוח באופן רנדומלי) קטן ממספר המקסימלי שניתן להשתמש במספר ספציפי זה אזי :נציב במיקום זה את המספר שהוגרל ונספור piecesCounter כך שנדע כמה מספרים ממנו יש לנו ונאפס את flags כי הצלחנו להציב מספר בתוך המטריצה וחוזרים ללולאה של העמודות בשביל לעבור לעמודה הבאה.

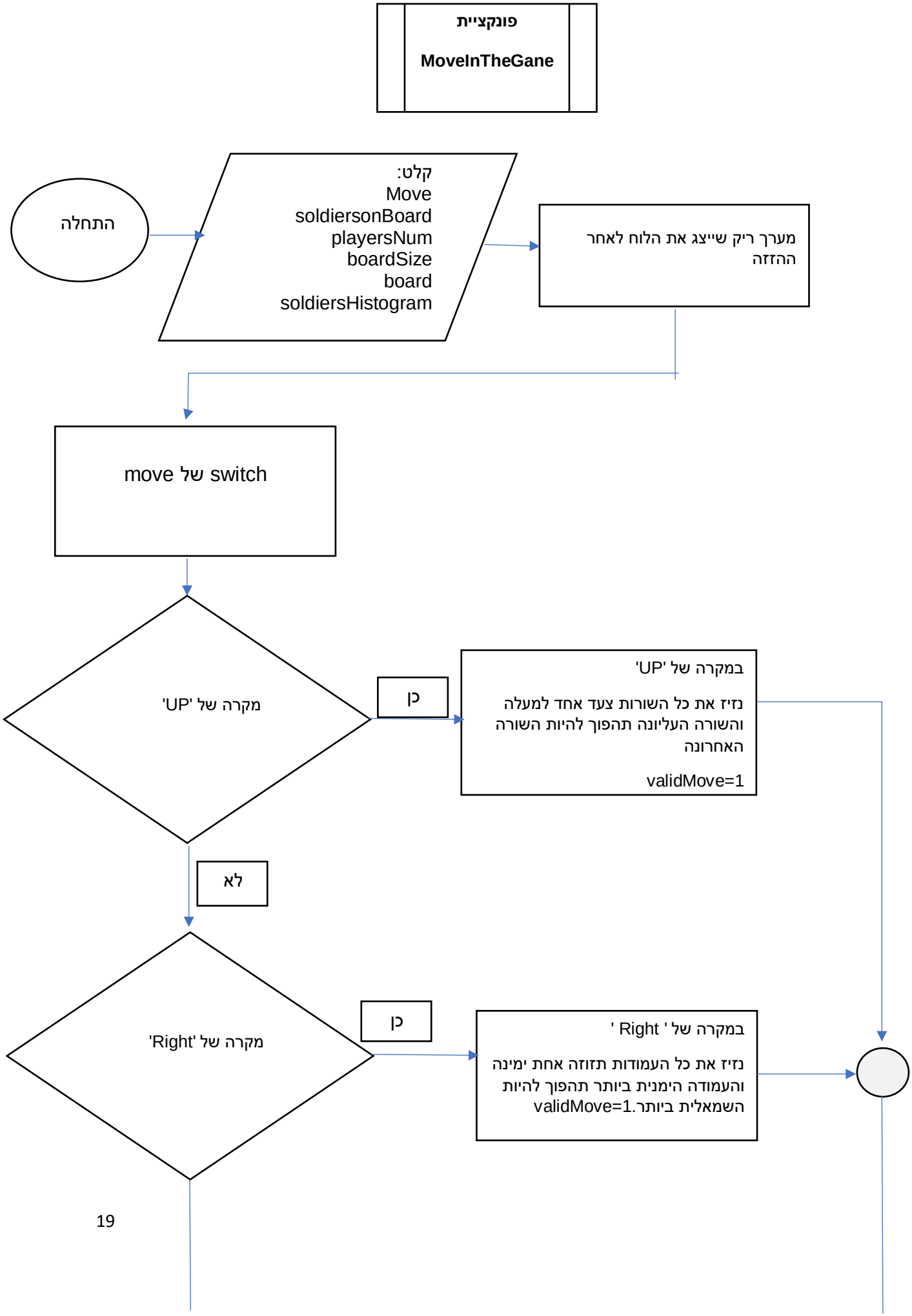
ייתכן מצב שהאלגוריתם יחלק מספרים על גבי הלוח בעזרת randi באופן שיצור מצב שהמספרים שנותרו לו להציב לא מתאימים בנקודה (i,j)*. והאלגוריתם יכנס ללולאה אינסופית בניסיון להציב מספרים לא אפשריים*.

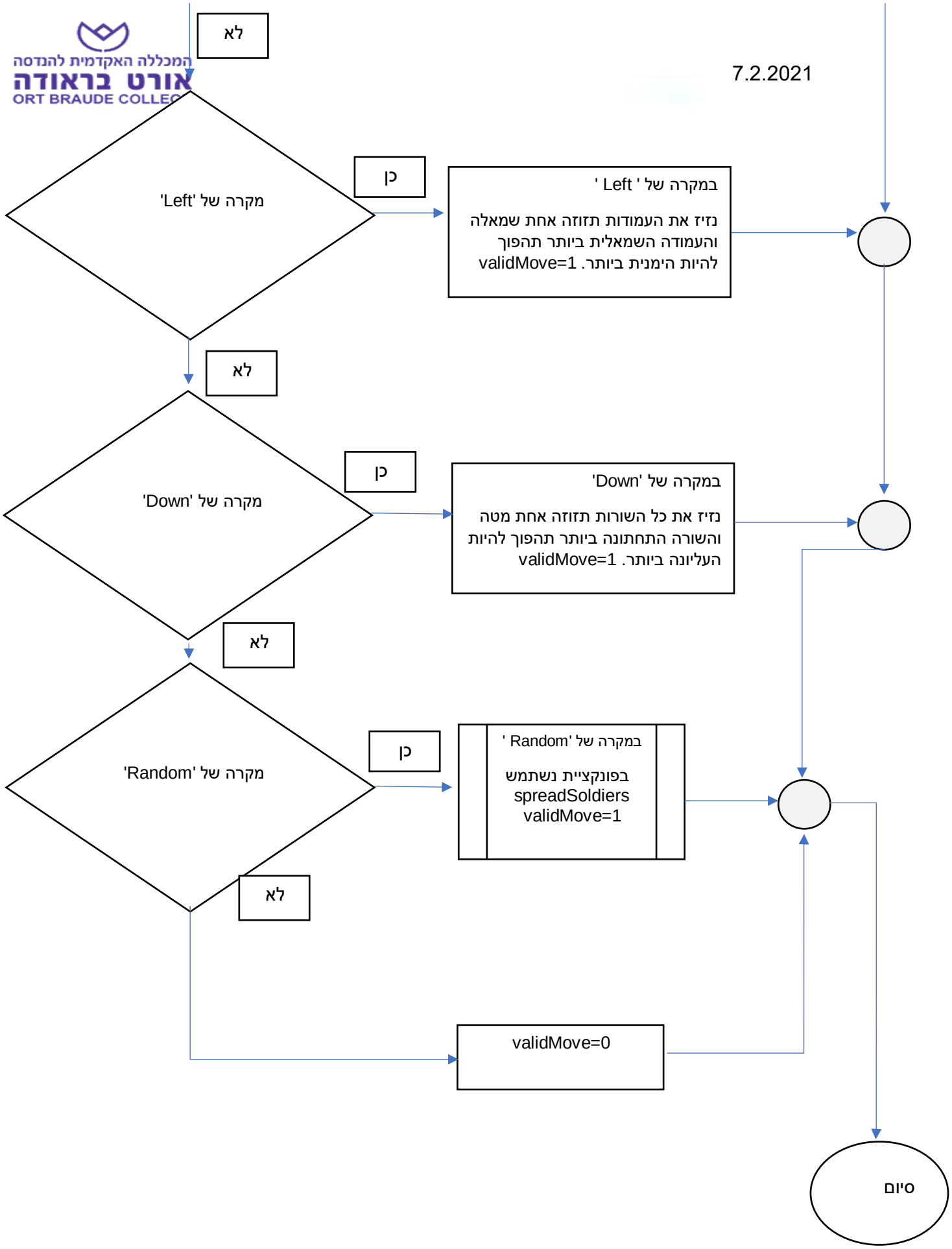
לכן אחד הרעיונות שעלו בדעתנו הם שבמצב זה נצטרך לנסות לחלק מחדש שוב מספרים רנדומליים עד שהאלגוריתם יצליח.

וכדי למנוע לולאה אינסופית נשתמש במערך flags כך שכל מספר שננסה נסמן שניסינו אותו ב flags כ-1 ואז אם איברי flags יהיו כולם אחדים זאת אומרת שניסינו כל מספר ובכל זאת לא היתה אופציה של הצבה לכן במצב זה checkConflict שווה 1 והוא יכנס למצב אתחול הלולאה while מחדש ככה שהוא יתחיל להציב מספרים רנדומליים מחדש.

מטרת הלולאה for לבדוק אם כל ערכי ה- flags הם אחד ואם כן checkConflict שווה לאחד וזה אומר שאנחנו נכנסים ללולאה אין סופית (אם לא תטופל) ואנחנו צריכים לטפל בזה אחרת (בעזרת אתחול הלולאות מחדש).

* מספרים לא אפשריים להצבה – לא ניתן לשים את אותו מספר בנקודה או מספר שהתבצע שימוש מקסימלי בו.


פונקציית
MoveInTheGane
התחלה
קלט:
Move
soldiersonBoard
playersNum
boardSize
board
soldiersHistogram
מערך ריק שייצג את הלוח לאחר ההזזה
move של switch
מקרה של 'UP'
כן
במקרה של 'UP'
נזיז את כל השורות צעד אחד למעלה והשורה העליונה תהפוך להיות השורה האחרונה
validMove=1
לא
מקרה של 'Right'
כן
במקרה של ' Right '
נזיז את כל העמודות תזוזה אחת ימינה והעמודה הימנית ביותר תהפוך להיות השמאלית ביותר.validMove=1


לא
מקרה של 'Left'
כן
במקרה של ' Left '
נזיז את העמודות תזוזה אחת שמאלה
והעמודה השמאלית ביותר תהפוך
להיות הימנית ביותר. validMove=1
לא
מקרה של 'Down'
כן
במקרה של 'Down'
נזיז את כל השורות תזוזה אחת מטה
והשורה התחתונה ביותר תהפוך להיות
העליונה ביותר. validMove=1
לא
מקרה של 'Random'
כן
במקרה של 'Random '
בפונקציית נשתמש
spreadSoldiers
validMove=1
לא
validMove=0
סיום

פונקציית MoveInTheGame

בפונקציה זו נציג לשחקן 5 אופציות של תזוזה של השחקנים על גבי הלוח.

בפונקציה זו השתמשנו במשפט תנאי של Switch כדי שעבור כל מקרה הלוח יבצע הזזות שונות.

נגדיר משתנה בשם validMove שתפקידו יהיה לבדוק האם הקלט של תזוזות השחקנים הינו תקין או שלא. במקרה והקלט יהיה תקין נפלוט שהוא שווה ל1 , במידה והקלט לא תקין נפלוט אותו כשווה ל-0.

עבור מקרה שהשחקן ירצה להזיז את כל השחקנים תזוזה אחת **מעלה** כל השורות יזוזו מעלה והשורה הראשונה תהפוך להיות השורה האחרונה.

עבור מקרה שהשחקן ירצה להזיז את השחקנים תזוזה אחת **מטה** כל השורות יזוז מטה והשורה האחרונה תהפוך להיות השורה הראשונה.

עבור מקרה שהשחקן ירצה להזיז את השחקנים תזוזה אחת **ימינה** כל העמדות יזוזו ימינה והעמודה הימנית ביותר תהפוך להיות השמאלית ביותר.

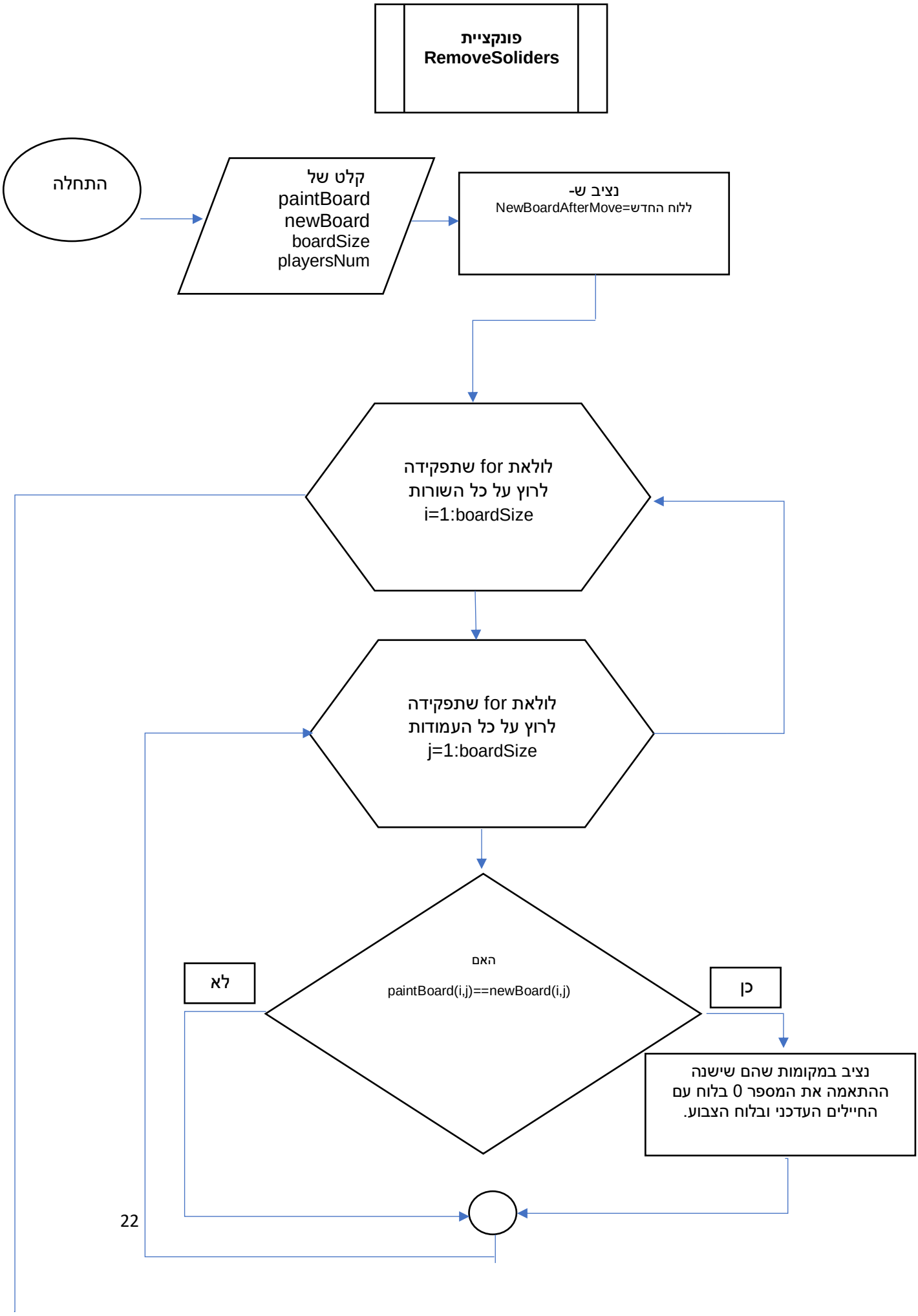
עבור מקרה שהשחקן ירצה להזיז את השחקנים תזוזה אחת **שמאלה** כל העמודות יזוזו שמאלה והעמודה השמאלית ביותר תהפוך להיות הימנית ביותר.

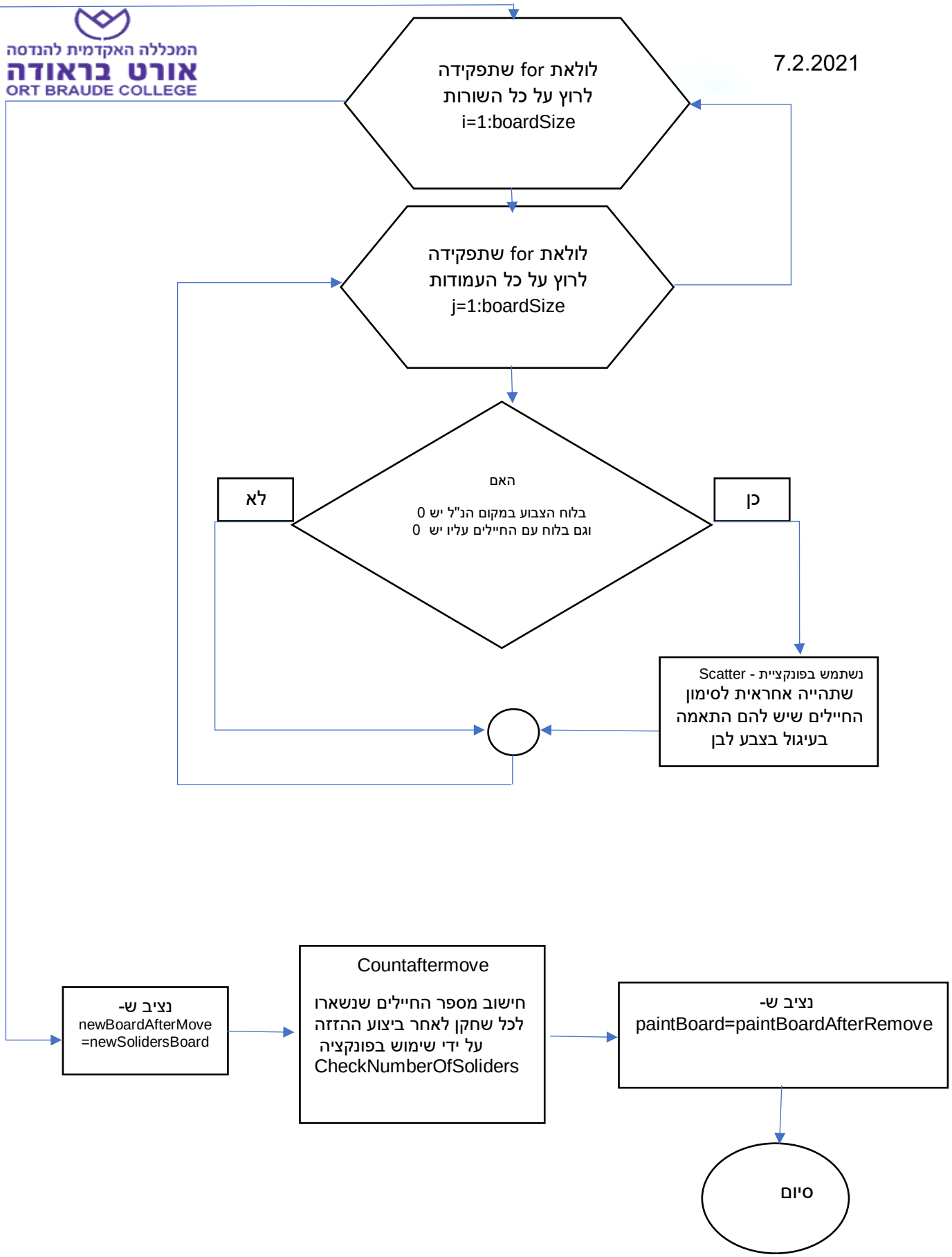
עבור מקרה שהשחקן ירצה לסדר את השחקנים מחדש באופן **אקראי** על גבי הלוח יבחר באופציה זו בידיעה שאף חייל לא ייפול על משבצת בצבע שלו.

בכל התזוזות הנ"ל הפלט validMove=1

עבר המקרה שהשחקן מזין **קלט לא תקין** נצא מן הפונקציה ותהייה לו האופציה לתקן את בהמשך.

ובמקרה הנ"ל הפלט יהיה validMove=0


פונקציית
RemoveSoliders
התחלה
קלט של
paintBoard
newBoard
boardSize
playersNum
נציב ש-
NewBoardAfterMove=ללוח החדש
לולאת for שתפקידה
לרוץ על כל השורות
i=1:boardSize
לולאת for שתפקידה
לרוץ על כל העמודות
j=1:boardSize
האם
paintBoard(i,j)==newBoard(i,j)
לא
כן
נציב במקומות שהם שישנה
ההתאמה את המספר 0 בלוח עם
החיילים העדכני ובלוח הצבוע.


לולאת for שתפקידה
לרוץ על כל השורות
i=1:boardSize
לולאת for שתפקידה
לרוץ על כל העמודות
j=1:boardSize
האם
בלוח הצבוע במקום הנ"ל יש 0
וגם בלוח עם החיילים עליו יש 0
לא
כן
נשתמש בפונקציית - Scatter
שתהייה אחראית לסימון
החיילים שיש להם התאמה
בעיגול בצבע לבן
נציב ש-
newBoardAfterMove
=newSolidersBoard
Countaftermove
חישוב מספר החיילים שנשארו
לכל שחקן לאחר ביצוע ההזזה
על ידי שימוש בפונקציה
CheckNumberOfSoliders
נציב ש-
paintBoard=paintBoardAfterRemove
סיום

פונקציית RemoveSoliders

פונקציה זו מקבלת 3 ערכים-(paintBoard,newBoard,boardSize,playerNum)
ותהייה אחראית על מחיקת החיילים אשר נמצאים על גבי המשבצות הצבועות בצבע הזהה להם
ובנוסף בכל מקום אשר מתקיימת התאמה זו נציב את הערך 0 אשר יסמן את הקובייה כריקה\לבנה.

נגדיר מטריצה חדשה שתקרא newBoardAfterMove שתהייה שווה למטריצה newBoard .

נעשה זו כדי לבצע את השינויים במטריצה החדשה ולא בזו שקיימת כבר בלוח.

נבצע זאת באמצעות 2 לולאות for אשר ירוצו מ1 עד מספר העמודות ומספר השורות בלוח וכאשר בין הלוח המקורי שהוא paintBoard לבין הלוח newBoard ישנה התאמה במקומות ה (i,j) אזי בלוח החדש newBoardAfterMove נציב במיקום זה את המספר 0. ובנוסף נציב בלוח paintBoard=0 על מנת לדאוג שגם צבע זה יהיה מוסר.

באמצעות 2 לולאות for נוספות שירוצו מ1 עד מספר השורות והעמדות נבדוק האם במקומות ה(i,j) יש לנו 0 בשתי הלוחות (לוח החיילים והלוח הצבוע) במידה וכן ניעזר בפונקציית scatter שתסמן לנו בעיגול לבן בתצוגה הגרפית את המקומות הללו לפני מחיקתם.

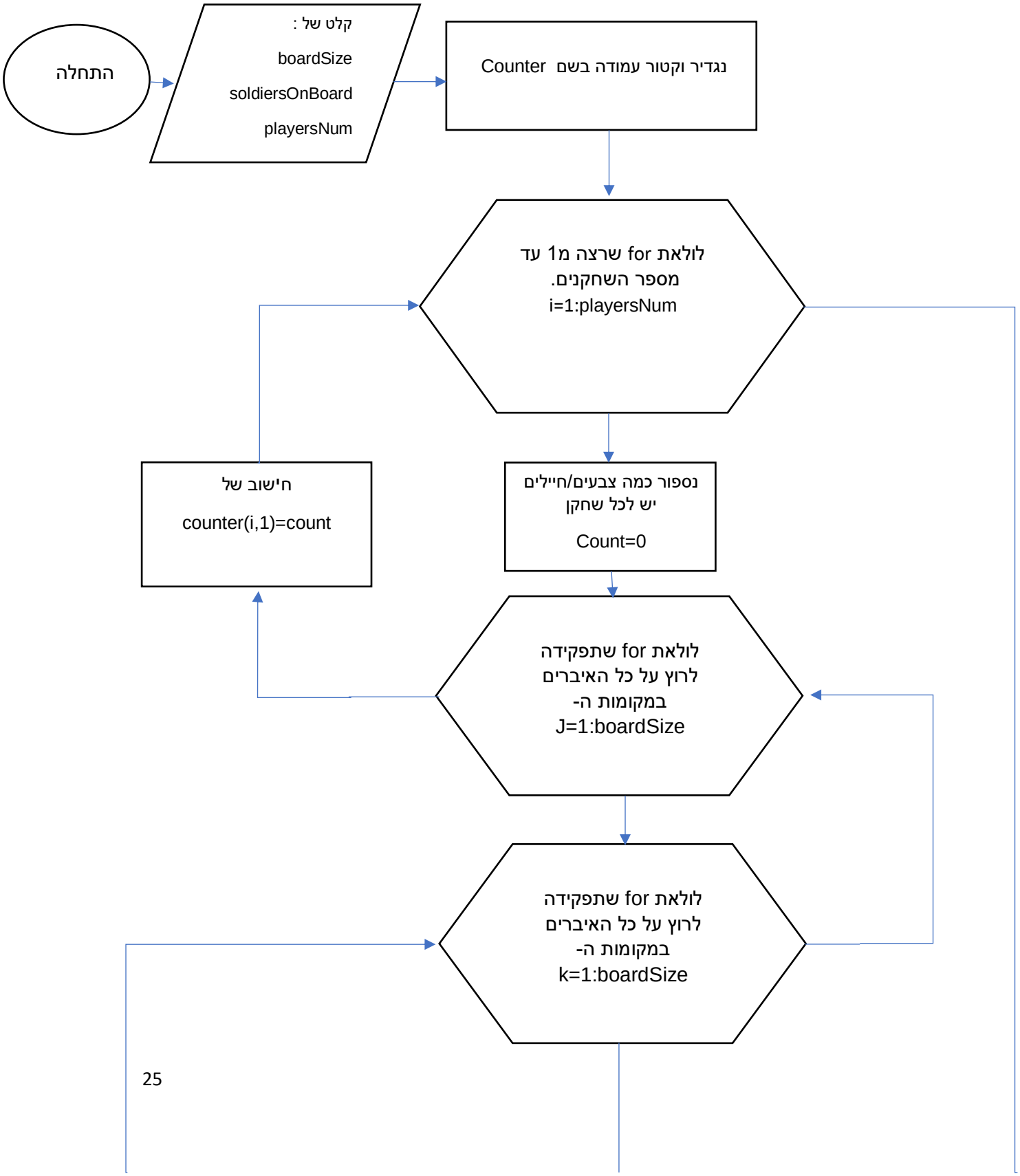
ולבסוף נגדיר את המשתנים:

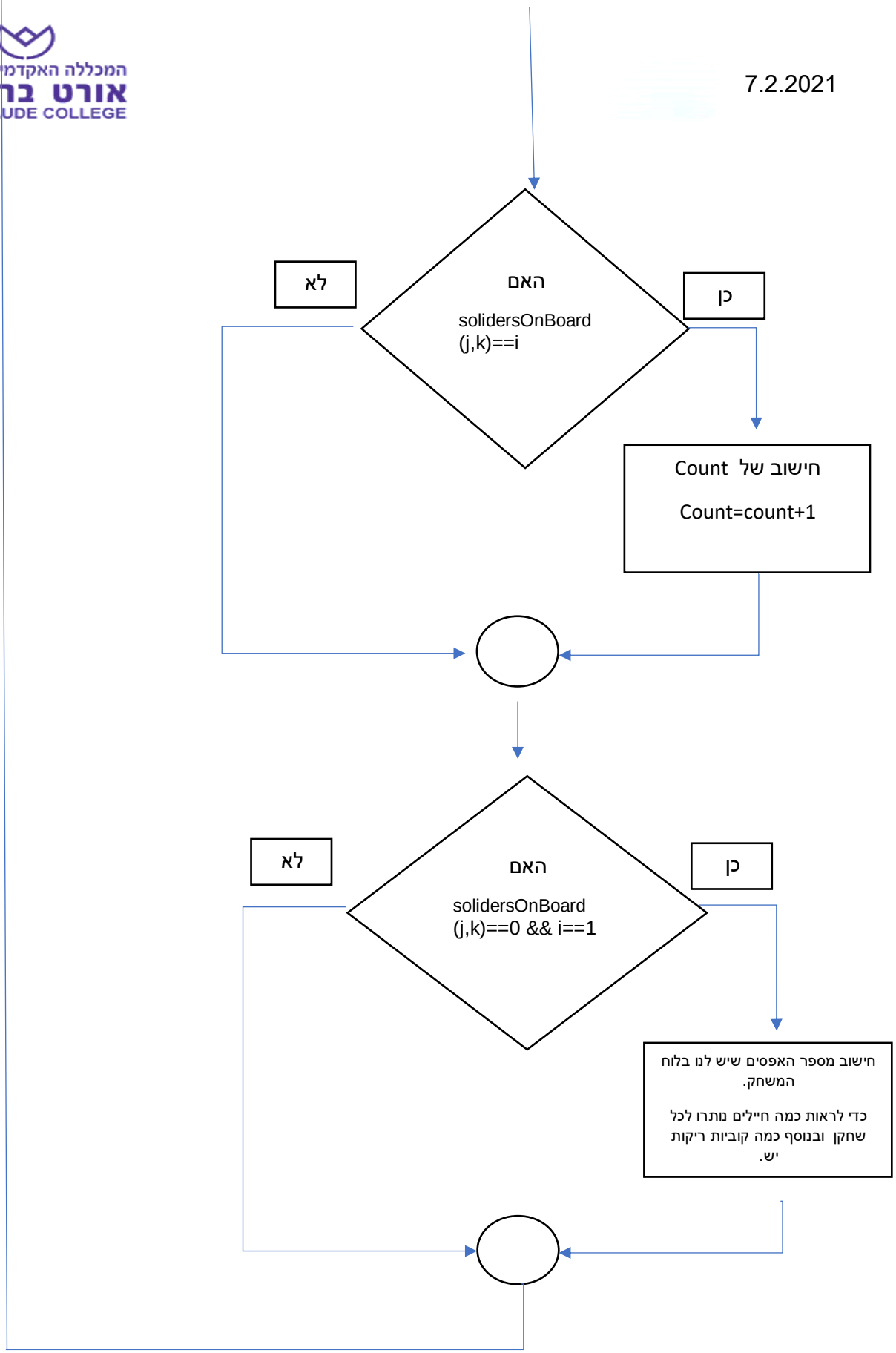
newSolidersBoard =newBoardAfterMove שזה בעצם מגדיר לנו את הלוח החדש עם החיילים עליו.

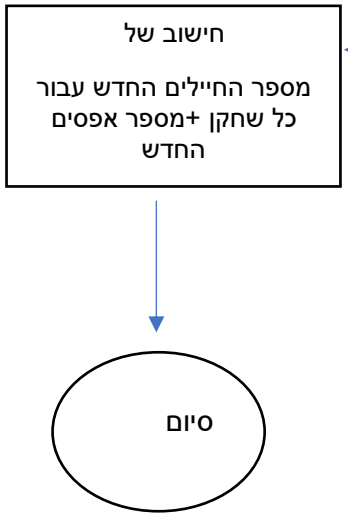
בשביל לבדוק כמה חיילים יש לנו עבור כל שחקן ניעזר בפונקציה CheckNumberOfSoliders ונפלוט זאת כמשתנה בשם Countaftermove.

paintBoardAfterRemove = paintBoard מגדיר לנו את הלוח החדש הצבוע.

פונקציית
CheckNumberOfSoliders




האם
solidersOnBoard
(j,k)==i
לא
כן
חישוב של Count
Count=count+1
האם
solidersOnBoard
(j,k)==0 && i==1
לא
כן
חישוב מספר האפסים שיש לנו בלוח המשחק.
כדי לראות כמה חיילים נותרו לכל שחקן ובנוסף כמה קוביות ריקות יש.



פונקציית CheckNumberOfSoliders

פונקציה זו תקבל ערכים של גודל הלוח ,את הלוח ומספר השחקנים ותפקידה יהיה לספור לאחר כל הזזה שנעשה על הלוח כמה חיילים נשארו לכל שחקן.

אופן הפעולה של פונקציה זו יהיה על ידי כך שנבנה מערך counter אשר תפקידו יהיה לאחסן את כל המידע על כמות חיילים לכל שחקן .

לאחר מכן נריץ לולאת for שתפקידה יהיה לרוץ על כל השחקנים כאשר בתוך לולאה זו יהיה לנו לולאה שתרוץ על כל השורות ועוד לולאה שתרוץ על כל העמודות ,ותצטרך לבדוק האם יתקיים שבמקום(j,k) שהוא שווה לאפס ועל מנת שלא תעשה הבדיקה הזו כמה פעמים נדרוש שזה יהיה שווה למספר 1 לדוגמא כדי שזה יבצע את הבדיקה באופן חד פעמי.

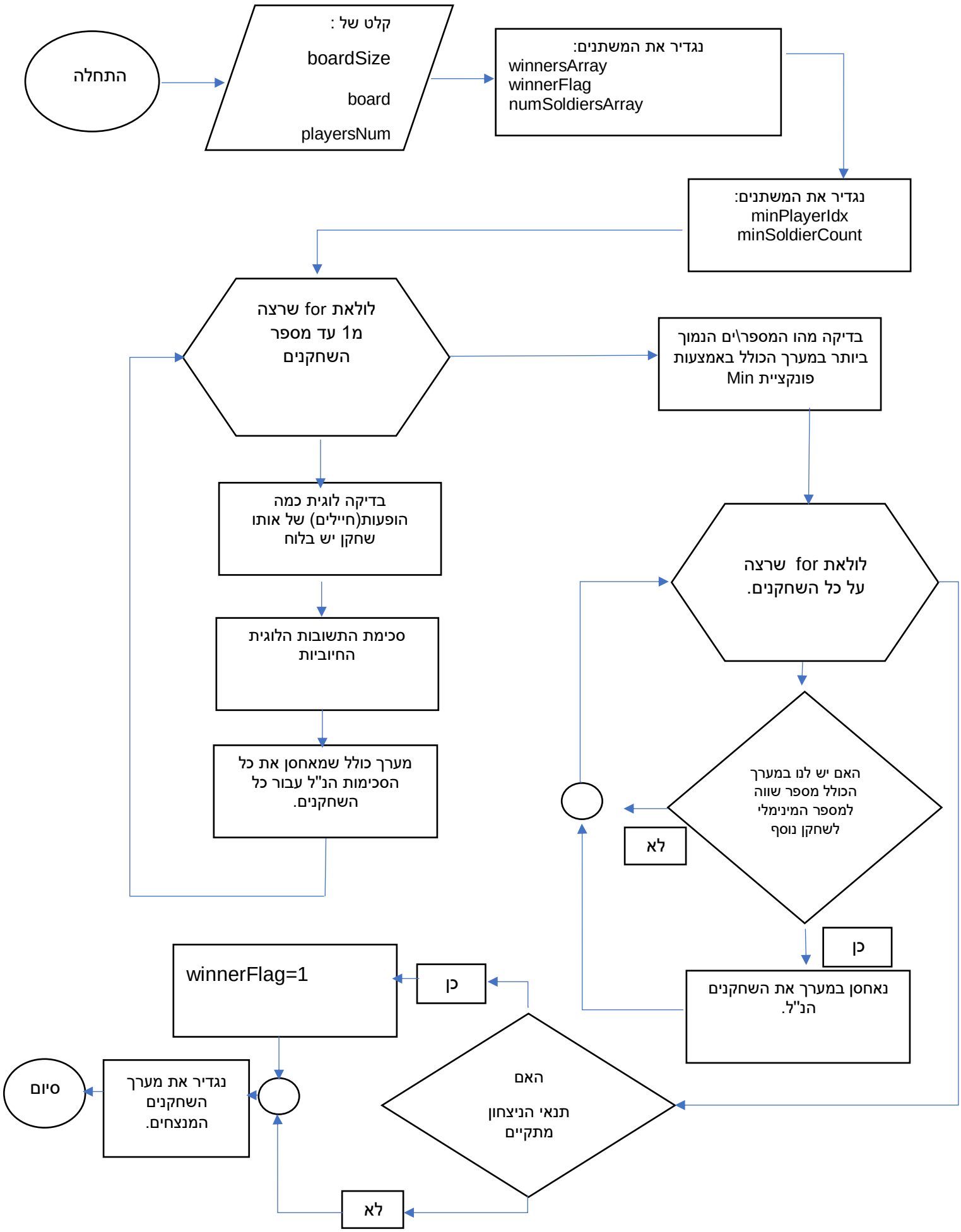
אם תנאי זה יתקיים אזי תכניס את האפסים החדשים שמצאנו לספירה וכך בעצם נדע כמה אפסים יש לנו.

לאחר מכן נדאג לעדכן את counter עבור כל שחקן כדי לדעת מה השינויים שנעשו עליו.
לאחר שנבצע את הבדיקה עבור כל השחקנים נדרוש לקבל בחזרה את counter וכך נדע מהו מספר החיילים החדש עבור כל שחקן ומהו מספר האפסים החדש ונפלוט זאת כמשתנה חדש בשם counterAfterMove .

המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

7.2.2021

פונקציית WinnerChecker



פונקציית WinnerChecker

בפונקציה זו נייצר ארבעה מערכים שתפקידם יהיה:

מערך ראשון יבדוק כמה חיילים יש לכל שחקן - numSoldiersArray .

ומערך שני תפקידו יהיה לאחסן איזה מספר שחקנים/שחקן הם בעלי מספר החיילים הנמוך ביותר- minPlayerIdx.

מערך שלישי יהיה אחסון מספר החיילים הנמוך ביותר שיש לשחקן\ שחקנים –minSolidersCount.

מערך רביעי יהיה מערך השחקן \שחקנים המנצחים - winnerArray.

נעזר בשני לולאות for שהראשונה תפקידה יהיה לבדוק על ידי חישוב לוגי את הלוח שנעשה בו שימוש אחרון על ידי השחקן האחרון וסכימת כל התשובות הלוגיות החיוביות עבור הצגת כמות החיילים שנותרו לכל שחקן כמערך -numSolidersArray.

לאחר מכן נבדוק מערך זה על ידי פונקציית min שתבדוק מי הוא השחקן בעל מספר החיילים הנמוך ביותר. עבור בדיקת מקרה של שוויון בין השחקנים נעזר בלולאה שתפקידה יהיה לבדוק האם יש במערך האחסון מספר זהה למספר הנמוך ביותר שמצאנו ובמידה וכן נציג כמערך את השחקנים בעלי מספר זה ונציג זאת כמספר האינדקס של שחקנים אלו -minPlayerIdx .

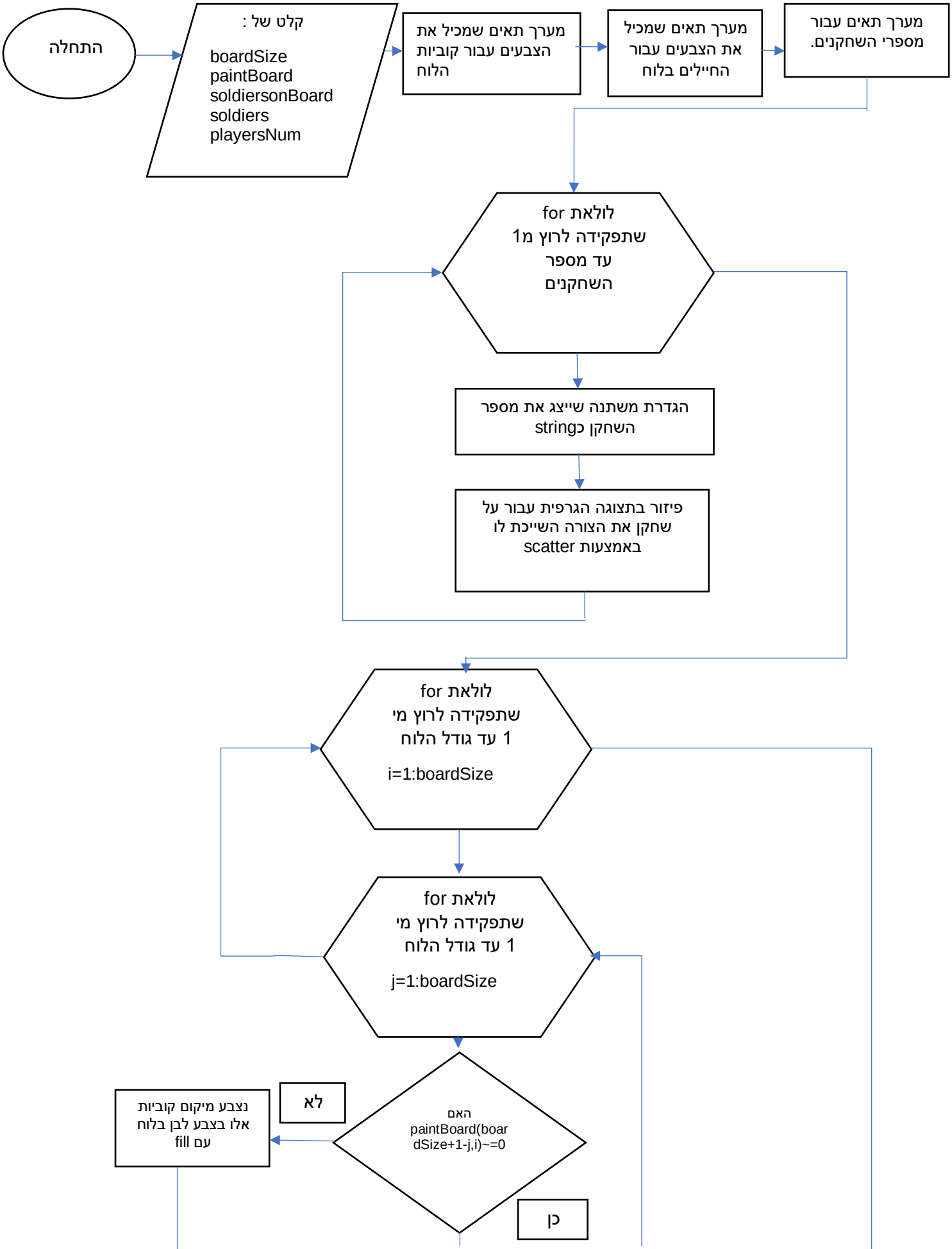
לאחר מכן נבדוק האם המערך של minSolidersCount מקיים את תנאי הניצחון עבור סיום המשחק(כאשר מספר החיילים קטן ממחצית מספר השורות או העמודות בלוח) , נעשה זאת על ידי בדיקה חד פעמית של האיבר הראשון במערך מכיוון שכל איברי המערך יהיו זהים. ואם כן נסמן זאת באמצעות winnerFlag=1

ולבסוף נפלוט את מערך השחקנים המנצחים – winnersArray.

המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

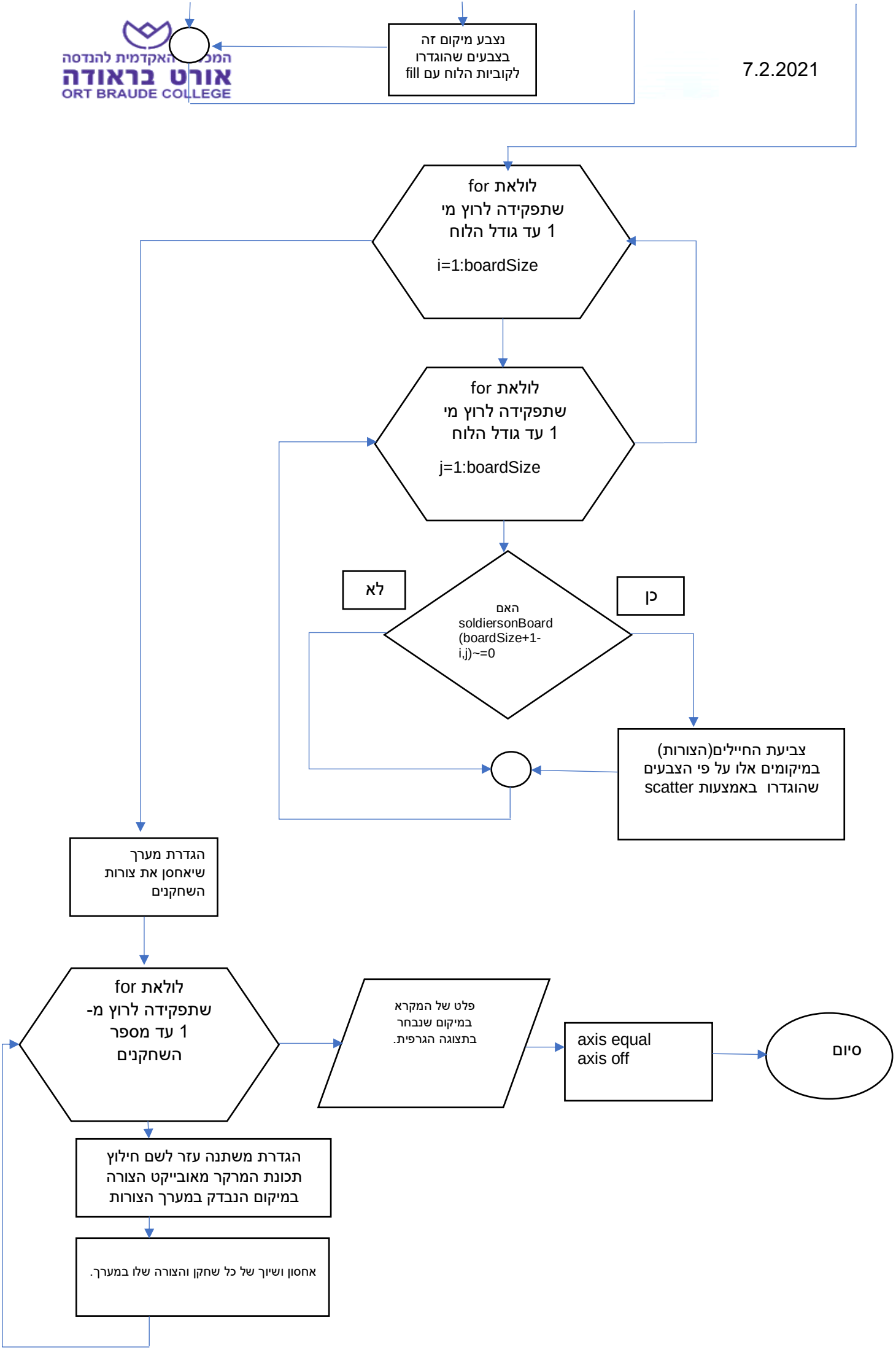
7.2.2021

פונקציית-PlotBoard



המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

7.2.2021



פונקציית-PlotBoard

פונקציה זו אחראית על התצוגה הגרפית של המשחק: בניית הלוח הנדרש וצביעת קוביות הלוח ובנוסף צביעת החיילים של כל השחקנים ולבסוף יצירת מקרא המשייך כל שחקן לצורה שלו ולצבע שלו במשחק.

נגדיר 2 מערכי תאים שיהיו אחראים על הצביעה:

1. boardColors אחראי על צביעת קוביות הלוח.
2. solidersColors אחראי על צביעת החיילים בלוח.

הגדרת מערך תאים שתפקידו לאחסן וליצור את שמות השחקנים והצורות שלהם בשם shapes .

נבצע את בניית הלוח\ צביעת הלוח על ידי 2 לולאות for שירוצו על גבי גודל הלוח בשורות והעמודות ובאמצעות תנאי הבדיקה של האם הלוח הצבוע שונה במיקום הנבדק שונה מ0 אזי נצבע את הקוביות בצבעים שנקבעו מ- boardColors .
במידה ותנאי הבדיקה שווה ל-0 נצבע את הקוביות שנותרו בלוח שיצרנו בצבע לבן.

באמצעות 2 לולאות for שרצות מ1 עד גודל הלוח בשורות והעמודות נבדוק האם החיילים שנמצאים על גבי הלוח במיקום הנבדק שונים מ-0 ואם כן נצבע את החייל הנבדק בצבעים של החיילים שנקבעו מ- solidersColors .

לאחר מכן נגדיר מערך שיהיה אחראי על אחסון צורות השחקנים , ניעזר בלולאת for שרצה מ1 עד מספר השחקנים ,ובתוכה נגדיר משתנה עזר לשם חילוץ תכונת המרקר מאובייקט הצורה במיקום הראשון שלו (מכיוון שמיקום זה מגדיר את הצורה עצמה של האובייקט) ונאחסן את כל הצורות במערך בשם handarray.

נייצר את המקרא של התצוגה הגרפית באמצעות legend של ה- handarray ונמקם אותו בלוח הצד הימני.

חלק ב'

פונקציית Game

```
clc
clear;
close all
%% Game General Settings
playersNum=6;
boardSize=7;

%% Generate paintBoard and soldiersBoard
soldiers=['o','x','p','s','d','^'];
[initcount,paintBoard]=InitBoard(playersNum,boardSize); %will output the initial board with equals colors in the game
[soldiersonBoard]=SpreadSoldiers(paintBoard,initcount,playersNum,boardSize);

% Initializing
countaftermove = initcount ; % countaftermove counts the floors for each player
winnerFlag   = 0;
winnersArray = 0;
playerTurn   = 0; %initial state 0,means player 1
round        = 0;

% display how many initial soldiers each player have.
for i= 1:playersNum
    fprintf('player %d has %d soldiers\n',i, countaftermove(i));
end

%% Game Flow
% Infinite loop until someone wins
while ~winnerFlag

    sprintf("Player %d turn\n",playerTurn+1);
    sprintf("Player %d turn\n",round+1);
```

```
    sprintf("Player %d turn\n",round+1);
    hold on

    title ( sprintf('Turn Player - %d Round %d' ,playerTurn+1,round+1));
    PlotBoard(boardSize,paintBoard,soldiersonBoard,soldiers,playersNum);

    fprintf("Player %d turn\n",playerTurn+1);

    %% The player chooses the move
    validMove=0;  %check for valid input move
    while validMove==0
        move=input('Choose your move(Up,Right,Left,Down,Random): ','s');
        [soldiersonBoard_temp,validMove]= MovesInTheGame(move,soldiersonBoard,playersNum,boardSize,paintBoard,countaftermove)
        if validMove==0
            disp('error, enter valid move: Up ,Right, Left ,Down,Random');
        else
            soldiersonBoard = soldiersonBoard_temp;
        end
    end

    %% Plot the board after the move
    PlotBoard(boardSize,paintBoard,soldiersonBoard,soldiers,playersNum)
    pause(3) %this pause to show the soliders on the board after move

    %% Marking and removing the soldiers-floor matching
    [soldiersonBoard,countaftermove,paintBoard]=RemoveSoliders(paintBoard,soldiersonBoard,boardSize,playersNum);
    pause(3) %this pause show the markers of the soliders will remove
    PlotBoard(boardSize,paintBoard,soldiersonBoard,soldiers,playersNum);
```

```
    %% Soldiers summary
    for i= 1:playersNum
        fprintf('player %d has %d soldiers\n',i, countaftermove(i));
    end
    hold off

    %% Check for Winner
    [winnerFlag,  winnersArray] = WinnerChecker(soldiersonBoard, playersNum, boardSize);

    playerTurn = rem(playerTurn+1, playersNum); % turn changing
    if playerTurn==0
        round=round+1;
    end
end

%% Winner Announce
disp("The Winner is player number: ");
disp(winnersArray)
```

פונקציית InitBoard

```
function [countPaintFloors,paintBoard]= InitBoard(playersNum,boardSize)
% The function is Creating a board and painting it

numberOfPieces=fix(boardSize^2/playersNum);      % determines how many soldiers/pieces on the board
board=randi([1 playersNum],boardSize,boardSize); % generates a board matrix
color_counter_vec=zeros(playersNum+1,1);

% counting colored squares and put zero in case of overflow
for i = 1:playersNum
    count=0;
    for j= 1:boardSize
        for k=1:boardSize
            if board(j,k)==i
                count=count+1;
            end
            if count>numberOfPieces % overflow case - put zero instead
                count=count-1;
                board(j,k)=0;
                color_counter_vec(playersNum+1,1)=color_counter_vec(playersNum+1,1)+1;
            end
        end
    end
    color_counter_vec(i,1)=count;
end


% color fixing to missed floors in board
for i = 1:playersNum
    checkZero=0; %check if we have too much zeros
```

```

% color fixing to missed floors in board
for i = 1:playersNum
    checkZero=0; %check if we have too much zeros
    if color_counter_vec(i,1)==numberOfPieces % checking current colored floors in color limit
        checkZero=1;
    end
    for j= 1:boardSize
        for k=1:boardSize
            if board(j,k)==0 && checkZero==0 %if square is not colored
                board(j,k)=i; %color square
                color_counter_vec(i,1)=color_counter_vec(i,1)+1; % add to count for each colored new square
                color_counter_vec(playersNum+1,1)=color_counter_vec(playersNum+1,1)-1;
                if color_counter_vec(i,1)==numberOfPieces % checking the color limit after adding new square
                    checkZero=1;
                end
            end
        end
    end
end


countPaintFloors=color_counter_vec;
paintBoard=board;
end
```

פונקציית SpreadSoldiers

```
function [boardWithSoldiers] = SpreadSoldiers(paintBoard,count,playersNum,boardSize)
%this function responsible to set up the soldiers on the board without any
%match with the paintboard

piecesBoard=zeros(boardSize); % soldiers matrix
piecesCounter=zeros(playersNum+1,1); % counts number of soldiers for each color includes zeros (white square)

flags=zeros(playersNum+1,1); % responsible to store all the try possibility numbers

checkConflict=0; % in case of conflict, recreating the original board
i=1;
while i<= boardSize
    j=1;
    while j<= boardSize

        while true % randomize soldiers until no conflict

            if checkConflict==1 % if confilct found need to randomize again
                % re-initialzing the loops
                i=1;
                j=1;
                piecesBoard=zeros(boardSize);
                flags=zeros(playersNum+1,1);
                checkConflict=0;
                piecesCounter=zeros(playersNum+1,1); % creates the counting array again with zeros
            end

            end

            % generate soldier number or nothing(0)
            randSoldierNum=randi([0,playersNum]);
            flags(randSoldierNum+1,1)=1; % sign this option for potential conflict

            % in case of generating no-soldier and valid soldier-floor constraint
            if randSoldierNum==0 && piecesCounter(playersNum+1,1)<count(playersNum+1,1)
                piecesBoard(i,j)=randSoldierNum;
                piecesCounter(playersNum+1,1)=piecesCounter(playersNum+1,1)+1;
                flags=zeros(playersNum+1,1);
                break;
            % in case of generating soldier check for:
                    % valid floor for the soldier
                    % color-soldier limit
            elseif  randSoldierNum~=0 && paintBoard(i,j)~=randSoldierNum && piecesCounter(randSoldierNum,1)<count(randSoldier
                piecesBoard(i,j)=randSoldierNum;
                piecesCounter(randSoldierNum,1)=piecesCounter(randSoldierNum,1)+1;          % counts how many times we used t
                flags=zeros(playersNum+1,1);
                break;
            end

            % in case of no match signs conflict
            checkConflict=1;
            for idx=1:playersNum+1
                checkConflict=checkConflict*flags(idx,1);
            end
        end
        j=j+1; % increase j for the loop
    end
    end
    i=i+1;  % increase i for the loop
end


boardWithSoldiers=piecesBoard;
end
```

פונקציית MovesInTheGame

```
function [newBoard,validMove]=MovesInTheGame(move,soldiersonBoard,playersNum,boardSize,board,solidersHistogram)
%this function responsible to move the soldiersonBoard after move cases

newBoard=[];%array that show us the board after some move
validMove=[];
switch move
    case 'Up'
        newBoard=[soldiersonBoard(2:end,:) ; soldiersonBoard(1,:)];
        validMove=1;
    case 'Right'
        newBoard=[soldiersonBoard(:,end) soldiersonBoard(:,(1:end-1))];
        validMove=1;
    case 'Left'
        newBoard=[soldiersonBoard(:,(2:end)) soldiersonBoard(:,1)];
        validMove=1;
    case 'Down'
        newBoard=[soldiersonBoard(end,:); soldiersonBoard(1:end-1,:)];
        validMove=1;
    case 'Random'
        newBoard=spreadSoldiers(board,solidersHistogram,playersNum,boardSize);
        validMove=1;


    otherwise  %this option active when the player input some other word diffrent from the options
        validMove=0;
end

end
```

פונקציית RemoveSoliders

```
function [newSolidersBoard,countaftermove,paintBoardAfterRemove] =RemoveSoliders(paintBoard,newBoard,boardSize,playersNum)
%this function will remove the soliders that have match after some turn
%and resposible to circle the soliders that will be removed
newBoardAfterMove=newBoard;


% checking for paintBoard and soldiers match, in match putting there zero in case
for i=1:boardSize
    for j=1:boardSize
        if paintBoard(i,j)==newBoard(i,j)
            newBoardAfterMove(i,j)=0;
            paintBoard(i,j)=0;
        end
    end
end

%circle the soliders that will remove with white color
for i=1:boardSize
    for j=1:boardSize
        if paintBoard(boardSize+1-i,j)==0 && newBoardAfterMove(boardSize+1-i,j)==0
            scatter(j,i,500,paintBoard(boardSize+1-i,j),'w','Marker','o','LineWidth',2);
            legend('off');
        end
    end
end
```

```
    end
end

newSolidersBoard=newBoardAfterMove; %the new sloiders board after the removes

countaftermove=CheckNumberOfSoldiers(boardSize,newBoardAfterMove,playersNum); %check how many soliders each player have after

paintBoardAfterRemove=paintBoard; %the new paintboard after the removes
end
```

פונקציית CheckNumberOfSoldiers

```
function counterAfterMove =CheckNumberOfSoldiers(boardSize,solidersOnBoard,playersNum)
% this fuction check how many soliders each player have after some move

counter=zeros(playersNum+1,1);
% counting how many soldiers left to the players
for i = 1:playersNum
    count=0;
    for j= 1:boardSize
        for k=1:boardSize
            if solidersOnBoard(j,k)==i
                count=count+1;
            end

            if solidersOnBoard(j,k)==0&&i==1
                counter(playersNum+1,1)=counter(playersNum+1,1)+1;
            end
        end
    end
    counter(i,1)=count;
end

counterAfterMove=counter;
end
```

פונקציית WinnerChecker

```
function [winnerFlag,  winnersArray] = WinnerChecker(board,playersNum, boardSize)
%this function will check which player\s have the smallest number of soliders
%and then check if the player\s exists the winner condition

winnersArray     = 0; % default - no winner yet
winnerFlag       = 0; % default - no winner yet
numSoldiersArray = [];
minPlayerIdx     = [];
minSoldierCount  = []; % min soldiers for player


% building histogram of soldiers
for i=1:playersNum
    occur_mat_i = board==i; % logic matrix of shows of player i
    occur_i= sum(sum(occur_mat_i));
    numSoldiersArray=[numSoldiersArray, occur_i];
end

% finding minimum soldiers
minSoldierCount = min(numSoldiersArray);
for i=1:playersNum
    if numSoldiersArray(i)==minSoldierCount
        minPlayerIdx=[minPlayerIdx,i];
    end
end
```

```
    end
end

%cheking winning condition
if minSoldierCount(1)<(boardSize/2)
    winnerFlag=1;
end

winnersArray = minPlayerIdx;

end
```

פונקציית PlotBoard

```
function [] = PlotBoard(boardSize,paintBoard,soldiersonBoard,soldiers,playersNum)
% the functions is responsbale to display the boared with the soliders on
% it with the legend

%% Color settings for board and soldiers
boardColors={[0.364 0.690 0.937],[0.996 0.5 0.5],[0.757 0.937 0.757],[0.996 0.597 0.898],[0.996 0.898 0.699],[0.398 0.398 0.3
soldiersColors={[0 0 0.79],[0.989 0 0],[0.160 0.741 0],[0.796 0 0.597],[0.699 0.464 0],[0.238 0.050 0.050]}; %colors of all t


%% Creating shapes legend
shapes = {};
for i= 1:playersNum
    player= ['player ',num2str(i)];
    shapes = [shapes, scatter(1,1,10,soldiersColors{i},soldiers(i),'DisplayName',player )];
end

% Plot the Board with colors
for i=1:boardSize
    for j=1:boardSize
        if paintBoard(boardSize+1-j,i)~=0 %check for not empty cube
            fill([i-0.5,i+0.5,i+0.5,i-0.5,i-0.5],...
                [j-0.5,j-0.5,j+0.5,j+0.5,j-0.5],...
                boardColors{paintBoard(boardSize+1-j,i)}); %paint the board and use the
        else
            fill([i-0.5,i+0.5,i+0.5,i-0.5,i-0.5],...
                [j-0.5,j-0.5,j+0.5,j+0.5,j-0.5],...
                'w');
        end
    end
end
```

```
% plot the colored soldiers on the board
for i=1:boardSize
    for j=1:boardSize
        if soldiersonBoard(boardSize+1-i,j)~=0
            scatter(j,i,200,soldiersColors{soldiersonBoard(boardSize+1-i,j)},'Marker',soldiers(soldiersonBoard(boardSize+1-i,
        end
    end
end

% markers shapes for legend
handarray =[]; %store all the shapes of the players
for i =1 : playersNum
    marker = shapes(i);
    handarray = [handarray, marker(1)];
end

legend(handarray,'Location','NortheastOutside'); %disp the shapes of the players in legend and the location for this legend

axis equal
axis off
end
```

חלק ג'

פונקציית InitBoard

שם המשתנה	תפקיד	מימדים	סוג
numberOfPieces	מחשב מהו מספר החיילים/קוביות צבועות לכל שחקן	סקלר	מספר
board	מטריצה שאחראית על הלוח עם הקוביות הצבועות	מטריצה	מספר
Color_conter_vec	סופר את הקוביות של הצבועות בלוח לאחר התיקונים	וקטור עמודה	מספר
count	לספור כמה צבעים ישלנו מכל צבע לפני תיקונים במקומות שיש יותר צבעים ממה שצריך	סקלר	מספר
checkZero	בודק האם יש לנו יותר מדי אפסים בלוח	סקלר	מספר
countPaintFloors	מציין את מספר הקוביות מכל צבע שנצבעו	וקטור עמודה	מספר
paintBoard	מציין את הלוח לאחר שנצבע	מטריצה	מספר
playersNum	מציין את מספר השחקנים	סקלר	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר

פונקציית SpreadSoliders

שם המשתנה	תפקיד	מימדים	סוג
piecesBoard	לוח שתפקידו לאגור את כל החיילים במקומות התקינים	מטריצה	מספר
piecesCounter	מערך שתפקידו לספור את מספר החיילים עבור כל שחקן	וקטור עמודה	מספר
flags	מערך שתפקידו לספור כל פעם את המספר שנספר ולאחסן אותו כמספר ספור	וקטור עמודה	מספר
checkConflict	אחראי למצב שיש לנו בו בעיה בבניית הלוח אז תפקידו לאפס ולייצר לוח מחדש	סקלר	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר
randSoldierNum	המספר המוגרל בפונקציית randi	סקלר	מספר
boardWithSoldiers	הלוח עם כל החיילים ממוקמים באופן הנדרש	מטריצה	מספר
paintBoard	הלוח שכבר נצבע	מטריצה	מספר
count	מערך שמאחסן את סכום הקוביות הצבועות שזה בעצם גם מספר החיילים	וקטור עמודה	מספר
playersNum	מספר השחקנים	סקלר	מספר

פונקציית -MovesInTheGame

שם המשתנה	תפקיד	מימדים	סוג
move	תפקידה להזיז את החיילים	-	טקסט
soldiersonBoard	מטריצה שמאחסנת את החיילים	מטריצה	מספר
playersNum	מספר השחקנים	סקלר	מספר
boardSize	גודל הלוח	סקלר	מספר
board	מציין את הלוח הצבוע	מטריצה	מספר
solidersHistogram	לספור כמה חיילים לכל שחקן	סקלר	מספר
newBoard	מייצג את הלוח לאחר הזזה	מטריצה	מספר
validMove	בדיקה האם קלט ההזזה תקין	סקלר	מספר

פונקציית RemoveSoliders

שם המשתנה	תפקיד	מימדים	סוג
playersNum	מציין את מספר השחקנים	סקלר	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר
newBoard	מייצג את הלוח לאחר הזזה	מטריצה	מספר
paintBoard	מייצג את הלוח הצבוע	מטריצה	מספר
newBoardAfterMove	מייצג את הלוח לאחר ההזזה	מטריצה	מספר
newSolidersBoard	מייצג את הלוח לאחר מחיקת החיילים המתאימים	מטריצה	מספר
countaftermove	בודק כמה חיילים נשאר לכל שחקן לאחר ההזזה	וקטור עמודה	מספר
paintBoardAfterRemove	מייצג את הלוח הצבוע לאחר מחיקת הצבעים המתאימים	מטריצה	מספר

פונקציית PlotBoard

שם המשתנה	תפקיד	מימדים	סוג
playersNum	מציין את מספר השחקנים	סקלר	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר
soldiersonBoard	מייצג את החיילים על גבי הלוח	מטריצה	מספר
paintBoard	מייצג את הלוח הצבוע	מטריצה	מספר
boardColors	מייצג את הצבע של קוביות הלוח	וקטור שורה	מספר
soldiersColors	מייצג את צביעת צורות החיילים בלוח	וקטור שורה	מספר
soldiers	מייצג את צורות החיילים	וקטור שורה	טקסט
shapes	מערך אובייקטים של צורות החיילים לבניית המקרא	וקטור שורה	טקסט\מספר
player	מייצג את השחקן המתאים במקרא	וקטור	טקסט
handarray	המערך לוקח מ- shapes את הצורות הגרפיות(מרקרים) ונשלח כפרמטר למקרא	וקטור	מספר
marker	משתנה עזר לשם חילוץ תכונת המרקר מאובייקט הצורה	וקטור	מספר

פונקציית WinnerChecker

שם המשתנה	תפקיד	מימדים	סוג
winnersArray	מערך עבור השחקנים המנצחים	וקטור	מספר
winnerFlag	סימון כאשר יש מנצח	סקלר	מספר
numSoldiersArray	מערך מספר החיילים לכל שחקן	וקטור	מספר
minPlayerIdx	מערך האינדקסים של השחקנים עם מספר החיילים המינימאלי	וקטור	מספר
minSoldierCount	מערך של מספר החיילים המינימלי	וקטור	מספר
playersNum	מציין את מספר השחקנים	וקטור	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר
board	מציין את הלוח הצבוע	מטריצה	מספר
occur_mat	משתנה שבודק את מספר ההופעות של אותו מספר במטריצה בצורה לוגית	מטריצה	מספר
occur_i	משתנה שסוכם את מספר ההופעות הלוגיות במטריצה	סקלר	מספר

פונקציית CheckNumberOfSoliders

שם המשתנה	תפקיד	מימדים	סוג
PlayersNum	מציין את מספר השחקנים	סקלר	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר
solidersOnBoard	מציין את הלוח עם החיילים עליו	מטריצה	מספר
counter	מערך של ספירת החיילים עבור כל שחקן	וקטור	מספר
counterAfterMove	מערך של ספירת החיילים עבור כל שחקן	וקטור	מספר

פונקציית Game

שם המשתנה	תפקיד	מימדים	סוג
PlayersNum	מציין את מספר השחקנים	סקלר	מספר
boardSize	מציין את גודל הלוח	סקלר	מספר
soldiers	מציין את צורות השחקנים	וקטור	מספר
round	מציין את מספר הסבב	סקלר	מספר
initcount	ספירת החיילים ההתחלתית עבור כל שחקן	וקטור	מספר
paintBoard	מציין את הלוח הצבוע כמטריצה	מטריצה	מספר
soldiersonBoard	מטריצה של החיילים על גבי הלוח	מטריצה	מספר
countaftermove	ספירת החיילים העדכנית לאחר התזוזות	וקטור	מספר
winnerFlag	סימון כאשר יש מנצח	סקלר	מספר
winnersArray	מערך של מספר השחקנים המנצחים	וקטור	מספר
playerTurn	מציין את מספר התור של כל שחקן	סקלר	מספר
validMove	משתנה שבודק האם הוזן תזוזה תקינה של השחקן	סקלר	מספר
soldiersonBoard_temp	מטריצה של החיילים על גבי הלוח (זמני כדי לבדוק גם אם המקרה לא תקין שהלוח יישמר)	מטריצה	מספר

חלק ד'

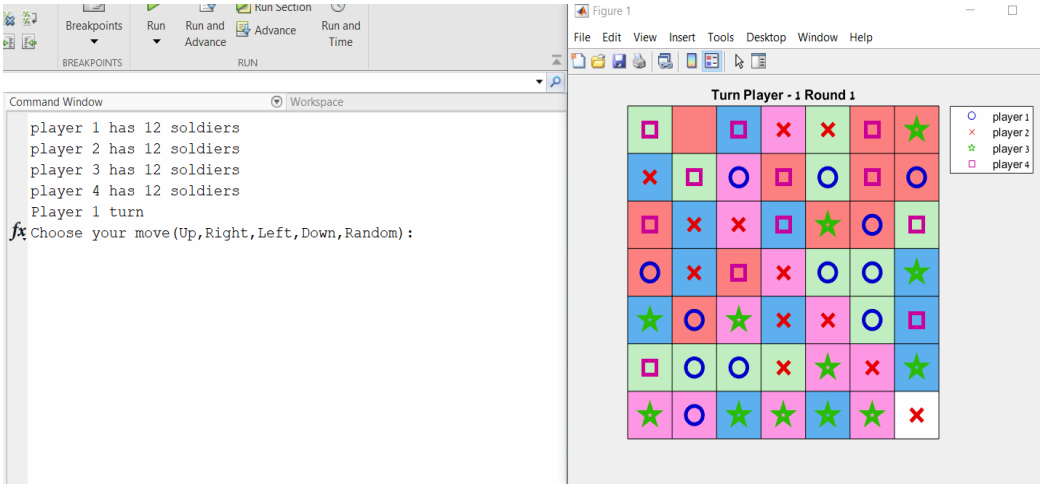
דוגמאות של צילומים מתוך המשחק:

דוגמא 1 :

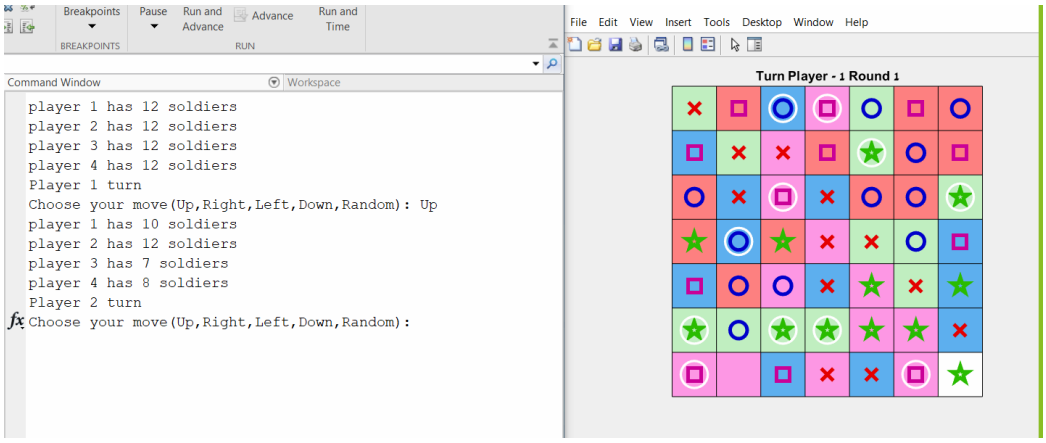
כמות שחקנים:4

לוח של: 7X7

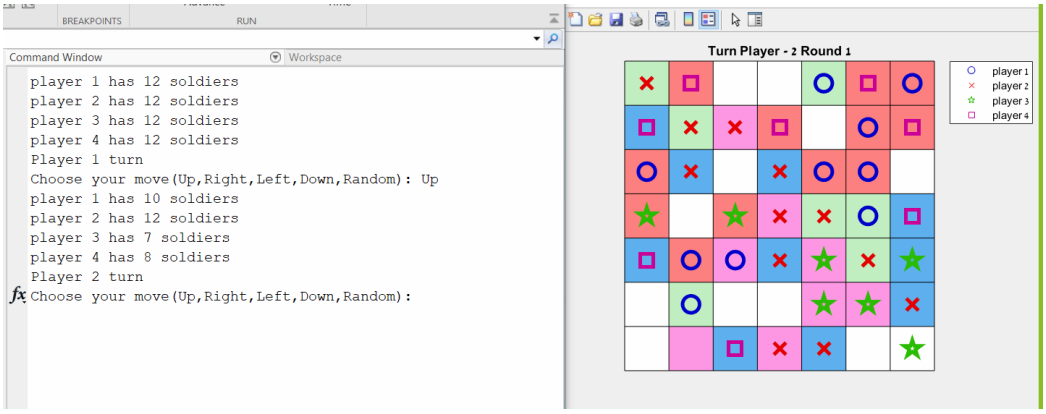
תחילת המשחק



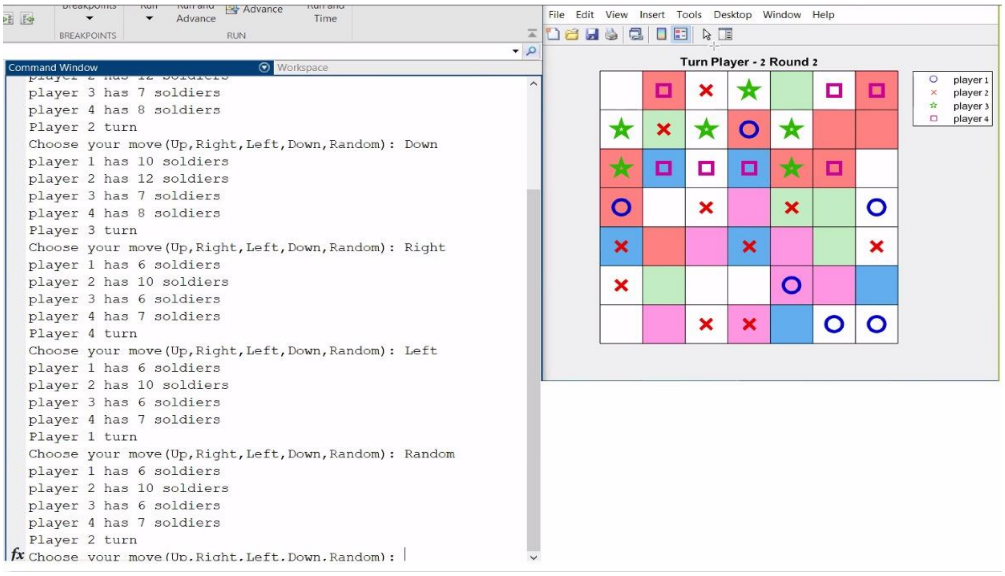
סימון החיילים שנמצאים על המשבצות שלהם



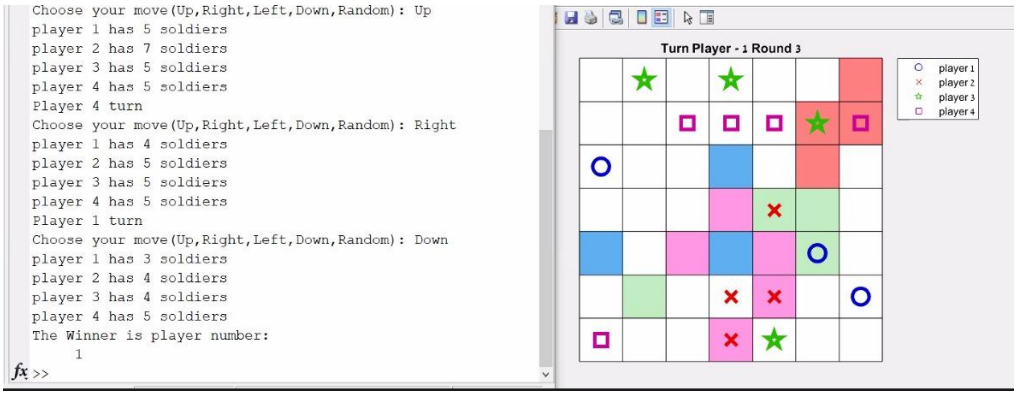
הסרת החיילים מחיקת הקוביות שהיו צבועות



מהלכים של כל התזוזות



מנצח והכרזה עליו

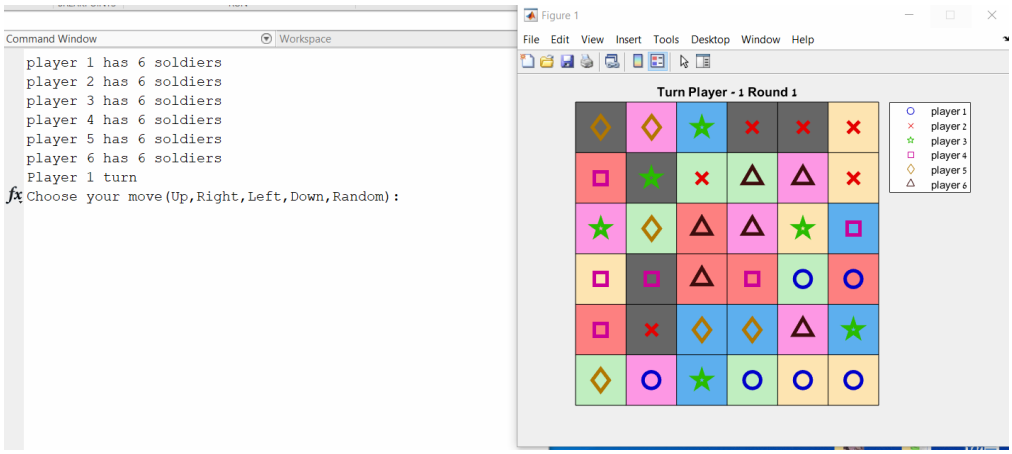


דוגמא 2:

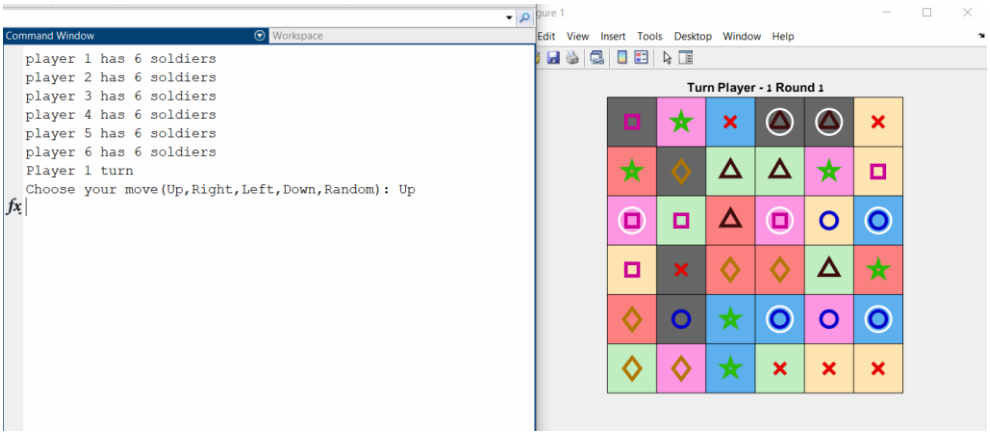
כמות שחקנים: 6

גודל הלוח:6X6

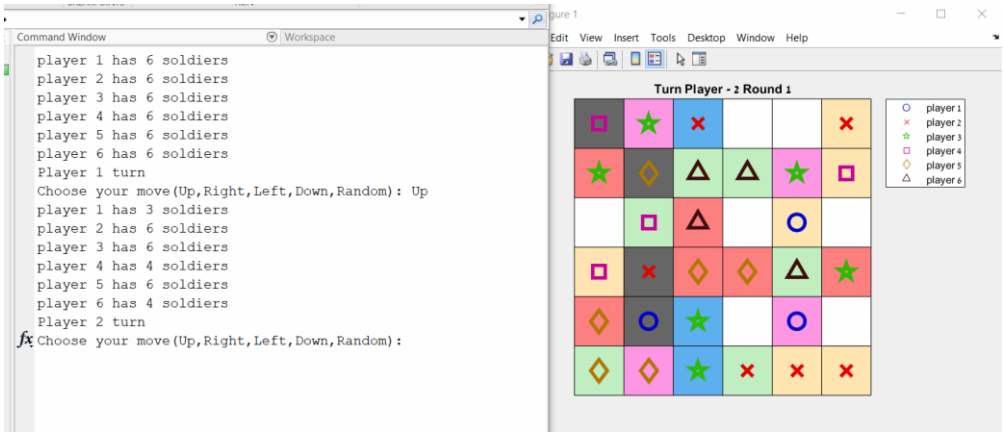
תחילת המשחק



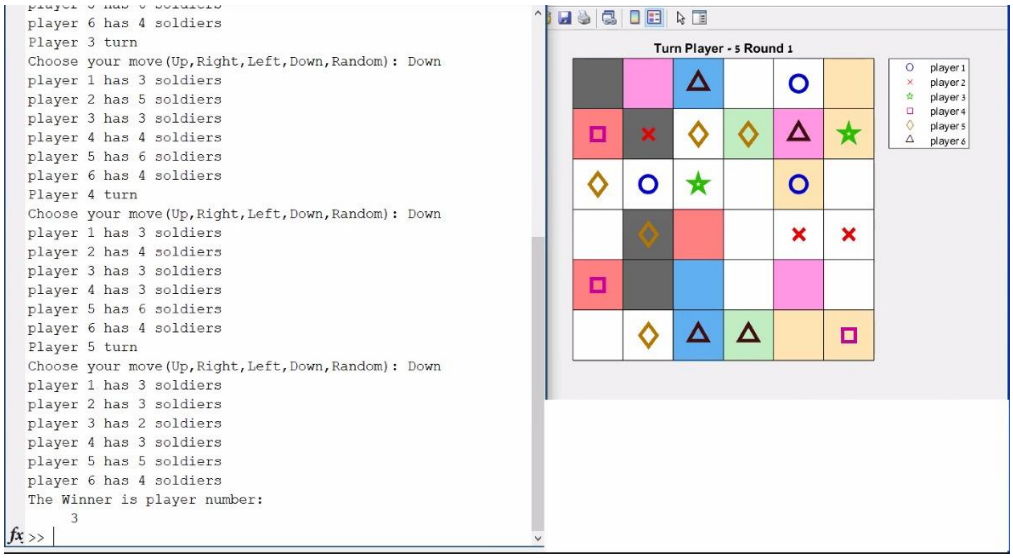
סימון החיילים שנמצאים על המשבצות שלהם לאחר תזוזה



הסרת החיילים מחיקת הקוביות שהיו צבועות



מנצח והכרזה עליו

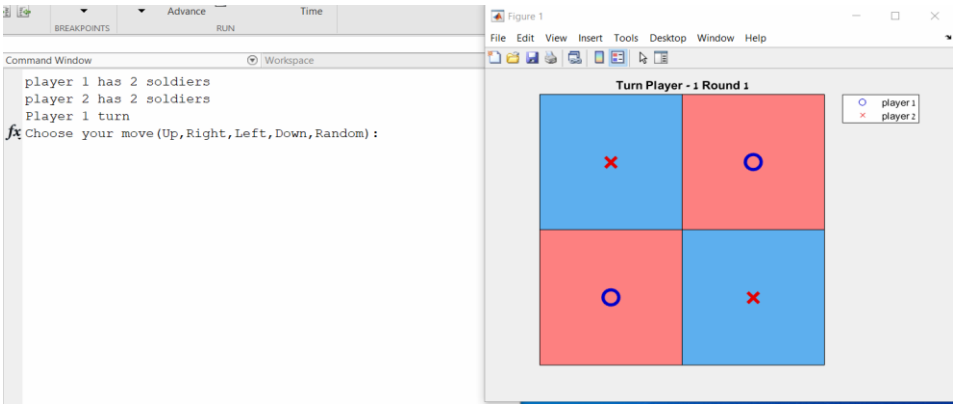


דוגמא 3

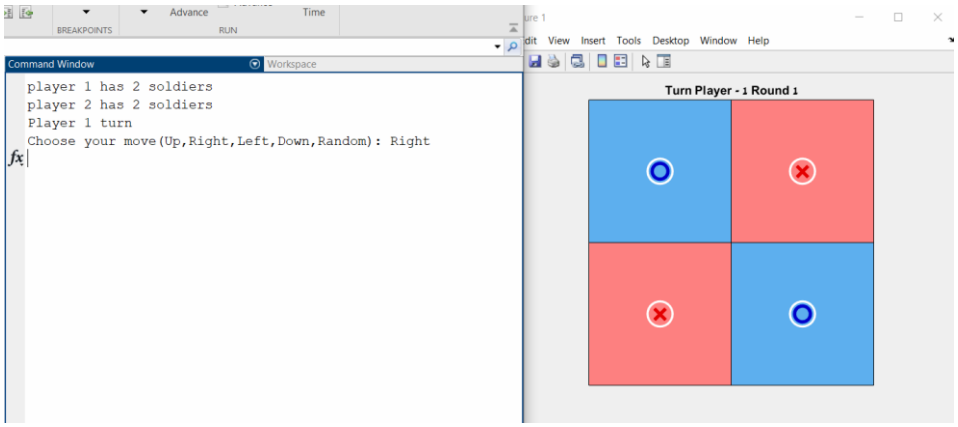
מספר השחקנים: 2

גודל לוח:2X2

תחילת המשחק

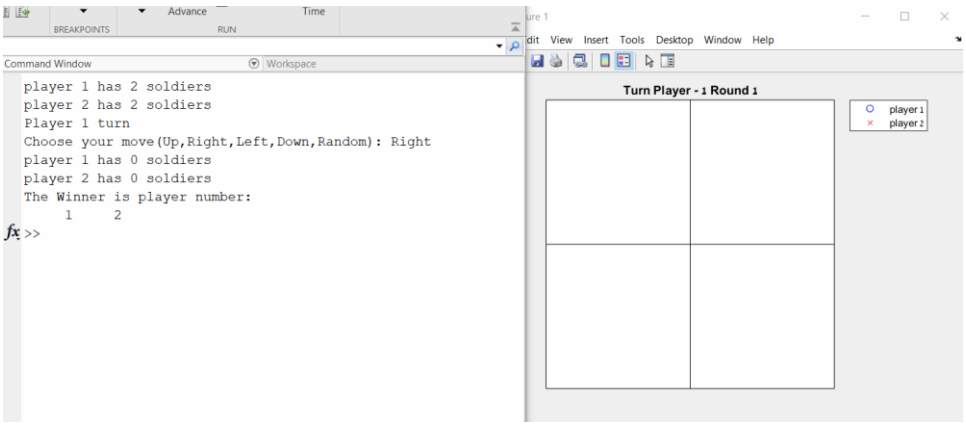


סימון החיילים שנמצאים על המשבצות שלהם לאחר תזוזה



הסרת החיילים מחיקת הקוביות שהיו צבועות

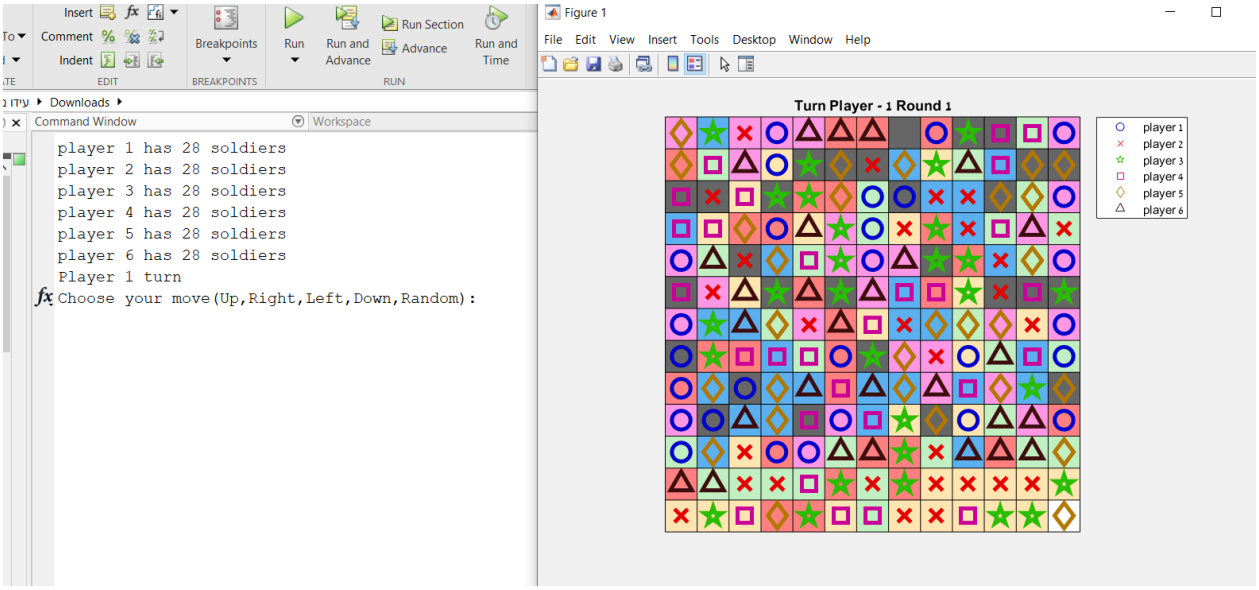
מנצח והכרזה עליו



דוגמה 4:

כמות שחקנים:6

גודל הלוח : 13X13

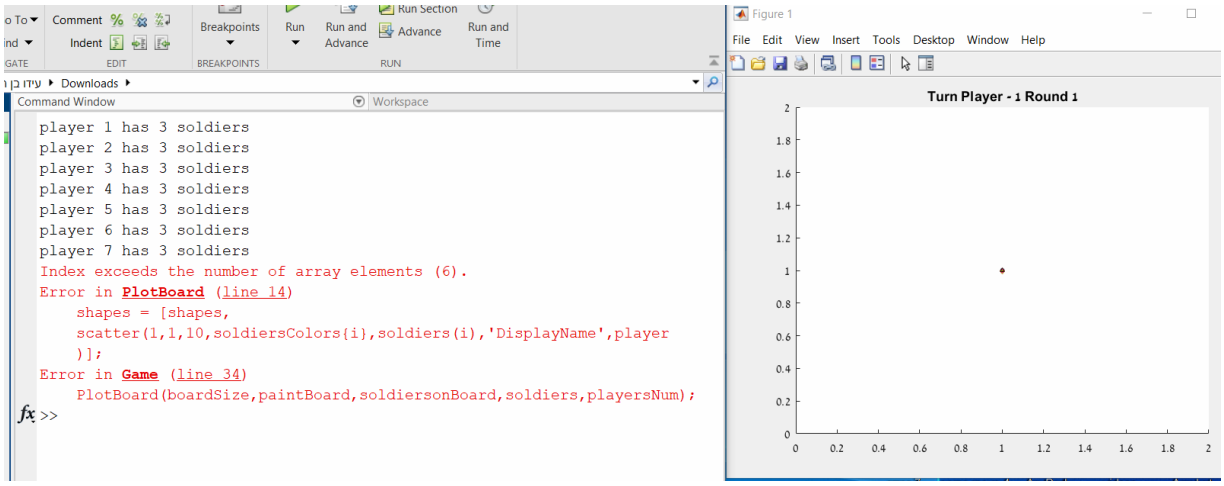


דוגמא 5 :

מצב לא אפשרי של מספר השחקנים יותר גדול ממספר המשבצות בלוח

כמות שחקנים:7

גודל לוח:5X5



ניתן לראות שהמשחק לא עובד מפני שאנו נתקלים בבעיה אשר מספר השחקנים יותר גדול מהמספר אשר הוגדר מראש כאשר ניתנו הוראות בניה של המשחק. מאחר והגדרנו עבור 6 שחקנים בלבד צורות אשר ייצגו אותם כאשר נזין יותר מ-6 שחקנים אנחנו נתקל בשגיאה

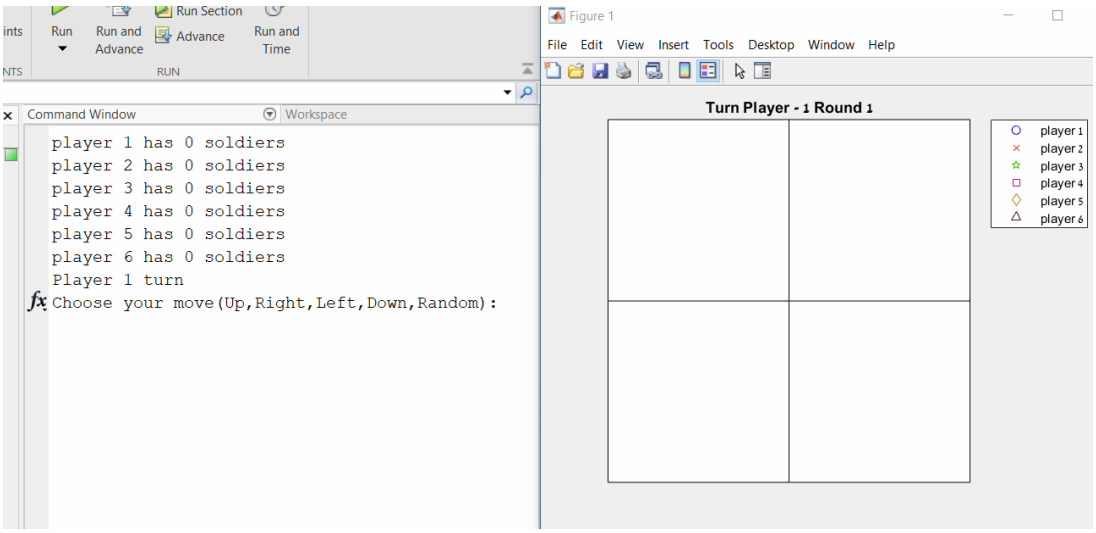
מפונקציית PlotBoard מפני שלא יהיה לה אפשרות לחלק לעוד שחקן צורה אשר לא מוגדרת מראש. לבסוף שגיאה זו תמנע מהמשחק להתחיל.

דוגמא 6:

מצב לא אפשרי : גודל של לוח קטן ביחס למספר השחקנים

כמות שחקנים:6

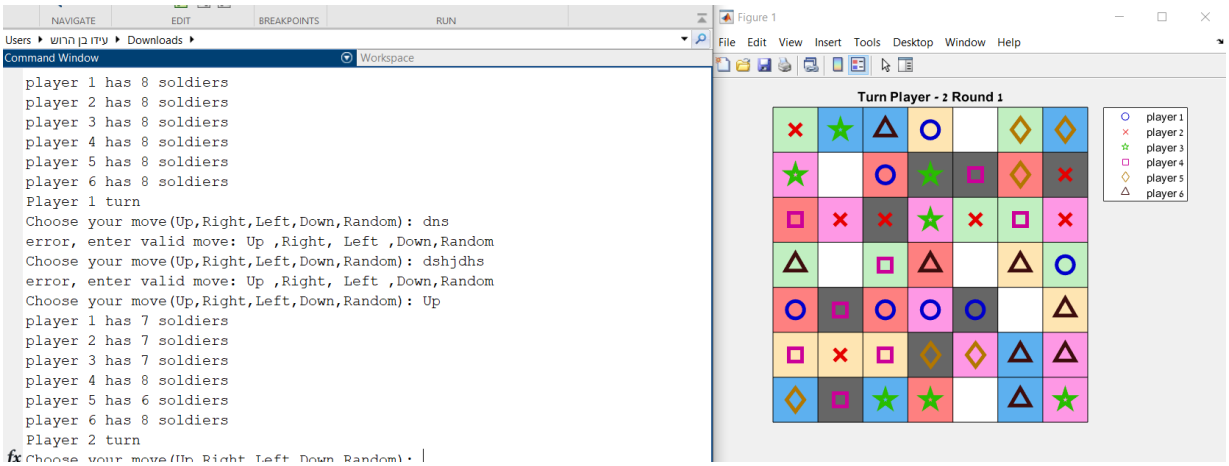
גודל לוח:2X2



כפי שניתן לראות המשחק נתקל בבעיה ואינו מציג שום חייל על גבי התצוגה מפני שאינו יכול לחלק מספר שווה של חיילים עבור כל המשתתפים לכן לא יהיה ניתן לשחק במשחק.

דוגמה 7:

מצב לא אפשרי : הזנת קלט לא תקין מהמבחר הקיים.



ניתן לראות שהמשתמש הזין קלט לא תקין עבור תזוזת החיילים בלוח ולאחר זיהוי קלט זה התוכנית מבקשת ממנו לתקן קלט זה מהמבחר תזוזות הקיים.

עזרה חיצונית לביצוע הפרויקט:

אתרים שנעזרנו בהם :

1.אופן שימוש ב- cell array במתלב (דוגמאות).

https://www.mathworks.com/matlabcentral/answers/608941-loop-over-an-array-or-list-of-vectors

2.בחירת גוונים של צבעים שלא ניתנו בהרצאה

https://www.w3schools.com/colors/colors_picker.asp?colorhex=F7CAC9

3.שילוב של פונקציית פיזור החיילים ופקודות random יחד:

הכוונה על ידי מרצה הקורס- ד"ר שאול סלומון בשעת קבלה בלחשוב על דרך לכלול את המקרה הפרטי של פיזור אקראי על גבי הלוח כאשר המשחק כבר מוכן מבלי ליצור פונקציה נוספת אשר תהיה אחראית על זה.

4.כיצד לגרום ללולאת while לעבוד כל עוד תנאי כלשהו מתקיים.

לאחר ששאלנו את מנחה המעבדה- מר יואב שלו איך ניתן לעשות זאת העלנו בפניו את האפשרות לרשום את המילה true לאחר שבדקנו בפורומים איך ניתן לעשות זאת קיבלנו הסבר ממנו על זה. בנוסף נעזרנו באתרים הנ"ל וראינו דוגמאות של איך משתמשים בזה.

https://www.mathworks.com/help/matlab/ref/while.html

https://www.mathworks.com/matlabcentral/answers/596284-how-to-use-while-true-loop

https://www.mathworks.com/matlabcentral/answers/596407-how-to-use-while-true-loop-for-this-qustion

https://www.mathworks.com/matlabcentral/answers/453403-how-can-i-get-a-infinite-loop-in-matlab

5.בפונקציית **spreadSoliders** מאחר ונתקלנו בבעיה של לולאה אינסופית שנוצרה מהמצב בו לפונקציית randi נשארו לה מספרים להצבה שלא מתאימים בנקודה(i,j) ולכן היא תמשיך לנסות להציב מספרים לא אפשריים.

קיבלנו הכוונה וקו מחשבה לבעיה של פתרון בעיה זו על ידי ניב בן הרוש בכך שהוא הסביר לנו שבשפת תכנות כאשר נתקלים בבעיות כאלה מצהירים על 'flag' .

לאחר חיפוש באינטרנט על משמעות של 'flag' מצאנו שאפשר לפתור בעיה זו על ידי הגדרת משתנה שכאשר מקרה זה מתקיים תפקידו יהיה להצהיר על כך ודאגנו שכאשר מצב זה קורה נדרש לאתחל את הלוח מחדש לניסיון חדש.

האתרים שנעזרנו בהם:

https://techterms.com/definition/flag

https://www.quora.com/What-is-a-flag-in-programming

/https://www.educba.com/matlab-flag

6.ביצוע הערות וחלוקה שלהם לטאבים באופן מסודר כפי שנדרש במטלה נעזרנו באתר:

https://www.mathworks.com/help/matlab/matlab_prog/comments.html

7.מיקום legend ב-Plot בצד ימין מחוץ ללוח נעזרנו:

https://www.mathworks.com/help/matlab/ref/legend.html#d122e731531