**תרגיל בית 3 – תיעוד חיצוני**

תיאור התכנית: תכנית זו תקבל מצב נתון בלוח דמקה ותחשב מהו המהלך הטוב ביותר לביצוע בשלב זה של המשחק עבור השחקן הרצוי. הצעד האופטימלי יחושב עבור מספר צעדים קדימה, שמתקבל כקלט לתוכנית. התכנית תחשב את הצעד האופטימלי בשיטת אלגוריתם minimax ותסרוק את כל אפשרויות המשחק של שחקן א', את התגובות של שחקן ב', לכל פעולה של שחקן א' את תגובותיו של א' לתגובותיו של ב' וכך הלאה עד שנגיע למספר הצעדים הנתון.

כל מצב בלוח שאליו נגיע לאחר רצף של מהלכים ייקבל ציון שישקף כמה המצב טוב עבור השחקן שאותו אנו בודקים (השחקן שמתקבל בתווית #Player ). הציון יחושב בעת הגעה למצב סופי כלשהו (מצב של ניצחון , הפסד או מספר צעדים מקסימלי) , וכל שחקן יבחר מבין כל המצבים האפשריים אליו יוכל להגיע את האופציה הטובה ביותר עבורו.

החישוב יתבצע תחת ההנחה שכל מהלך שיבצע כל אחד מהשחקנים הוא הטוב ביותר עבורו, והגרוע ביותר עבור היריב שלו וכך בכל שלב. לאחר שניתן ציון למצבים הסופים האפשריים (אופן חישוב הציון יוסבר בהמשך) בחירת המהלך הטוב ביותר תתבצע באופן הבא:

השחקן שעבורו אנו מחפשים את המהלך המיטבי ( השחקן שנמצא בתווית #Player) ייבחר מבין כל מהלכיו האפשריים את המהלך עם הציון המקסימלי.

לעומת זאת, השחקן שאינו השחקן שאנו בודקים עבורו את המהלך המיטבי ייבחר מבין כל מהלכיו האפשריים את המהלך עם הציון המינימלי, שכן זהו המצב הגרוע ביותר עבור השחקן הראשון והטוב ביותר בעבורו (הציון מחושב ביחס לשחקן הראשון).

כאשר נחזור מהקריאה הרקורסיבית האחרונה, ייבחר השחקן שאנו בודקים את המהלך עם הציון המקסימלי, ובכך נקבל את המהלך המיטבי עבורו.

קלט התוכנית:

התכנית תקבל כקלט את התוויות הבאות:

**#Board** - תווית המציינת מערך מילים ותכיל את לוח הדמקה. מערך המילים יהיה בגודל וכל מילה בו תקבל את אחד משלושת הערכים הבאים:

1 – מייצג כלי משחק לבן.

2 – מייצג כלי משחק שחור.

0 – מייצג תא ריק בלוח.

**#Player** - תווית שתייצג את השחקן עבורו נצטרך לחשב את המהלך המיטבי ותכיל את אחד מהערכים הבאים:

1 – החישוב צריך להתבצע עבור השחקן הלבן.

2 – החישוב צריך להתבצע עבור השחקן השחור.

**#Steps** - תווית שתייצג את כמות הצעדים המקסימלית שעבורה נצטרך לחשב את הצעד האופטימלי ("עומק הבדיקה").

בנוסף לתוויות הנ"ל נקבל גם את הכתובות של תוויות אליהן נרשום את הפלט.

פלט התוכנית:

פלט התכנית יירשם ל-3 כתובות שונות בזיכרון אותן נקבל כפרמטרים בפונקציית המשחק הראשית.

התוויות אליהן ירשם הפלט הן:

**#SrcPos**– כתובת בזיכרון אליה נרשום את האינדקס בו נמצא הכלי עבורו יתבצע המהלך הטוב ביותר.

**#DstPos** - כתובת בזיכרון אליה נרשום את אינדקס היעד אליו הכלי שצויין באינדקס המקור צריך לזוז בכדי לבצע את המהלך הטוב ביותר.

**#WinParam** – כתובת בזיכרון אליה ירשם הפרמטר המיטבי אותו חישבנו. ערך זה יחושב באופן הבא:

* 20 עבור ניצחון של השחקן עבורו בדקנו את המהלך הטוב ביותר (השחקן שערכו מופיע בתווית #Player ) המושג בכמות צעדים שערכה נמצא בתווית #Steps.
* -20 עבור הפסד של השחקן עבורו בדקנו את המהלך הטוב ביותר (השחקן שערכו מופיע בתווית #Player ) המושג בכמות צעדים שערכה נמצא בתווית #Steps.
* כל ערך אחר ייצג את ההפרש בין מספר הכלים של השחקן עבורו בדקנו למספר הכלים של השחקן היריב.

ערכים שליליים: ייצגו מצב בו הצעד האופטימלי יוביל לחיסרון מספרי בכלי השחקן שעבורו בדקנו צעד זה.

ערכים חיוביים: ייצגו מצב בו הצעד האופטימלי יוביל ליתרון מספרי בכלי השחקן שעבורו בדקנו צעד זה.

**פרוצדורות(שגרות)**

**getMaxWP**

שגרה זו מהווה שגרת מעטפת לשגרה הרקורסיבית שתבצע את חישוב המהלך האופטימלי.

קלט: השגרה תקבל כקלט במחסנית שלוש תוויות אליהן יכתבו הערכים שייצגו את המהלך האופטימלי: #SrcPos, #DstPos ו#Winparam. בנוסף תקבל שלוש תוויות בשטח משותף: #Steps, #Player ו #Board שישמשו לבדיקת המהלך.

פלט: השגרה תכתוב לתוויות שקיבלה במחסנית את הערכים הבאים: לתווית #SrcPos תרשום את אינדקס המקור של הכלי עבורו יתבצע המהלך הטוב ביותר, לתווית #DstPos תרשום את אינדקס היעד אליו צריך לזוז הכלי בכדי לבצע את המהלך הטוב ביותר ולתווית #WinParam את הפרמטר המיטבי שחושב כפי שהוסבר למעלה.

השגרה מכינה את המחסנית לקראת הקריאה הרקורסיבית הראשונה לשגרה getWPRec. בחזרה מהרקורסיה מבצעת המרה של כתובות של תאים בלוח, שחושבו עבור המהלך הטוב ביותר, לאינדקסים וכותבת לתוך התווית #WinParam את הערך המיטבי שחושב.

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| #WinParam | 2(sp) |
| #DstPos | 4(sp) |
| #SrcPos | 6(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה:

**getWPRec**

שגרה זו מחשבת את הערכים המיטביים עבור מהלך אופטימלי.

קלט: שגרה זו תקבל את הקלט במחסנית שתכיל ערכים המתאימים:

1. WP ראשוני - יאותחל לערך (100-) כאשר מחפשים מקסימום, ויאותחל לערך 100 כאשר מחפשים מינימום. בכך נאפשר שכל ערך ראשוני שיוחזר מהקריאה הרקורסיבית יעדכן את הערך המיטבי.
2. minOrMax – מציין האם בקריאה זו מחפשים מינימום או מקסימום. עבור הקריאה הראשונה של הפונקציה ערך זה יאותחל ל- 1 שכן השחקן הנבדק הוא זה שמחפש את המקסימום. עבור הקריאות הבאות נשנה בכל פעם את ערך זה מ (-1) ל – (1) ולהיפך.
3. שחקן נוכחי – השחקן שעבורו אנו מחשבים את המהלך המיטבי.
4. Steps – ערך המייצג את מספר הצעדים לבדיקה ("עומק הבדיקה").

פלט:

1. כתובת מקור – כתובת המייצגת את המקום בלוח בו נמצא הכלי שצריך לזוז.
2. כתובת היעד – כתובת המייצגת את המקום שאליו הכלי צריך לזוז.
3. ערך מיטבי – ערך ה Winning Parameter עבור הלוח הנתון והשחקן הנתון.

השגרה ממומשת באופן רקורסיבי והיא נעצרת ע"י אחד מהתנאים הבאים:

* אחד השחקנים ניצח – במקרה זה השגרה תבדוק האם המנצח הוא השחקן שאנו בודקים. אם כן תחזיר 20, אחרת תחזיר (-20).
* מספר הצעדים שנותרו לבדיקה הוא 0 – במקרה זה השגרה תקרא לשגרה calcWP שתחשב את הערך המיטבי כפי שיתואר בהמשך.

לאחר תנאי העצירה השגרה בודקת האם לשחקן הנוכחי יש צעדים חוקיים לביצוע. במידה ואין מתבצע דילוג על תורו. אחרת, הפונקציה תסרוק את הלוח עד שתמצא כלי התואם את צבע השחקן הנוכחי, ותבדוק עבורו את כל כיווני ההתקדמות האפשריים. כאשר יימצא כיוון התקדמות אפשרי השגרה תקרא לאחת מהשגרות WhiteMov או BlackMov בהתאם לצבע השחקן. שגרות אלו יעדכנו את הלוח בהתאם כלומר יבצעו את ההזזה של הכלי ולאחר מכן תקראנה ל – getWPRec . נמשיך כך עד שניעצר ע"י תנאי העצירה. בחזרה מתנאי העצירה תתבצענה ההשוואות הבאות לפרמטרים של השגרה הקוראת:

* האם השגרה הקוראת (caller) מחפשת מקסימום או מינימום.
* האם ערך ה WP שחושב בשגרה הנקראת (calle) עדיף על פני זה של השגרה הקוראת.

במידת הצורך השגרה תעדכן את ערכי ה – WP , SrcPos, DstPos של השגרה הקוראת.

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| מספר צעדים שנותרו לבדיקה | 2(sp) |
| ערך שחקן נוכחי | 4(sp) |
| מינימום או מקסימום (1 או 1-) | 6(sp) |
| ערך WinParam ראשוני (100 או 100-) | 10(sp) |
| מקום לערך SrcPos | 12(sp) |
| מקום לערך DstPos | 14(sp) |

מבנה המחסנית בקריאה לפונקציה:

**checkWin**

שגרה הבודקת עבור לוח נתון ושחקן נתון האם השחקן ניצח.

קלט: כתובת של התווית המכילה את הלוח במצב הנוכחי ומספר השחקן הנוכחי.

פלט : 1 עבור ניצחון של השחקן הנוכחי, אחרת 1-. הפלט יוחזר בראש המחסנית.

שגרה זו מחשבת ניצחון עבור השחקן הנוכחי באופן הבא: השגרה קוראת לשגרה countWB שסופרת את הכלים השחורים והלבנים שנמצאים על הלוח במצב הנתון (ומחזירה אותם בראש המחסנית). לאחר מכן מתבצעת בדיקה אם לשחקן היריב אין כלים על הלוח, אם כן תחזיר 1 כפלט, אם לא תבדוק את מספר הכלים של השחקן הנוכחי ואם הוא 0 תחזיר 1- כפלט. לאחר מכן, אם מספר הכלים של שני השחקנים שונה מאפס ולאף אחד מהשחקנים אין אפשרות לביצוע מהלך חוקי השגרה תבדוק אם מספר הכלים של השחקן הנוכחי גדול משל היריב ואם כן אז הוא ניצח ויוחזר 1 כפלט.

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| מקום לפלט | 2(sp) |
| ערך שחקן נוכחי | 4(sp) |
| #Board | 6(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה:

c**alcWP**

שגרה זו מחשבת את הערך המיטבי בהתאם למצב הנתון של הלוח ולשחקן אותו בודקים.

קלט: כתובת של התווית המכילה את הלוח במצב הנוכחי ומספר השחקן הנבדק.

פלט: הציון (WP) המתאים ללוח שהתקבל, יוחזר ברגיסטר R4.

השגרה קוראת לשגרה countWB שמחזירה את מספר הכלים מכל צבע. לאחר הקריאה מתבצע החישוב של מספר הכלים של השחקן הנבדק פחות מספר הכלים של השחקן היריב, אשר מוחזר כפלט.

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| ערך השחקן הנוכחי | 2(sp) |
| #Board | 4(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה:

**minOrMax**

שגרה זו בודקת האם השחקן הנוכחי הוא השחקן המופיע בתווית #Player. אם כן מחזירה 1 ואם לא מחזירה 1-.

קלט:

1. ערך השחקן הנוכחי.
2. כתובת התווית של השחקן אותו בודקים #Player .

פלט: 1- אם מחפשים ערך מינימלי, 1 אם מחפשים ערך מקסימלי. יוחזר במחסנית**.**

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| #Player | 2(sp) |
| ערך השחקן הנוכחי | 4(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה:

**movPiece**

שגרה זו מקבלת אינדקס מקור ואינדקס יעד של חייל על הלוח ומזיזה את החייל (מעדכנת את הלוח בהתאם).

קלט:

1. אינדקס מקור של חייל.
2. אינדקס יעד של חייל.

הקלטים יועברו inlineבעזרת שימוש ברגיסטר R5השומר את כתובת החזרה. כתובת של התווית המכילה את הלוח במצב נוכחי , תועבר במחסנית.

פלט: הפונקציה תעדכן את הלוח שהועבר אליה בתווית.

|  |  |
| --- | --- |
| ערך רגיסטר R5 | sp |
| #Board | 2(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה:

**פונקציות עזר:**

**WhiteMov, Blackmov**

שגרות אלה יהוו חלק משגרת המשחק הראשית. שגרות אלה יקראו כאשר נמצא מהלך חוקי עבור כלי מסוים. שגרות אלה יעדכנו את הלוח למצב שלאחר ביצוע הצעד ויכינו את המחסנית לקראת קריאה לפונקציה הרקורסיבית getWPRec. בהכנה לקריאה הרקורסיבית תתבצענה הפעולות הבאות:

* גיבוי הרגיסטרים.
* הכנסת מינימום/מקסימום למחסנית באופן הפוך לקריאה האחרונה.
* אתחול ערך WP ראשוני ל -100 כאשר מחפשים מקסימום ול- 100 כאשר מחפשים מינימום.
* החלפת שחקן נוכחי.
* השמה של מספר צעדים חדש שקטן באחד ממספר הצעדים הקודם.

לאחר החזרה מהקריאה הרקורסיבית של הפונקציה getWPRec יתבצעו הפעולות הבאות:

* עדכון הלוח בחזרה למצב הקודם שלפני התזוזה.
* שחזור הרגיסטרים.
* עדכון ערכי ה- WP, SrcPos ו - DstPos במידת הצורך.

קלט: כתובת אליה יש להזיז את הכלי, כתובת של החייל הנאכל, אם לא התבצעה אכילה יועבר 0.

פלט: הפונקציה תעדכן את הערכים הרלוונטיים אם נמצאו טובים יותר אך לא תחזיר פלט כלשהו.

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| #Player | 2(sp) |
| ערך השחקן הנוכחי | 4(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה:

**countWB**

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| #Board | 2(sp) |
| מקום פלט של השחקנים השחורים | 4(sp) |
| מקום פלט של השחקנים הלבנים | 6(sp) |
| ערכים קודמים שהיו במחסנית |  |

שגרה זו תספור את כמות הכלים השחורים והלבנים שנמצאים על הלוח המועבר אליה כקלט.

קלט: כתובת המכילה את המצב הנוכחי, מועברת במחסנית.

פלט: ערך מספר הכלים השחורים ומספר הכלים הלבנים הנמצאים על הלוח הנתון, יועברו בראש המחסנית.

מבנה המחסנית בקריאה לפונקציה:

**hasMove**

שגרה זו תבדוק האם לשחקן המועבר אליה כקלט יש לפחות מהלך אחד חוקי לביצוע.

קלט: השחקן עבורו רוצים לבדוק אם קיימים מהלכים חוקיים.

פלט: 1 אם קיים מהלך חוקי כלשהו, 0 אם לא קיימים מהלכים חוקיים, יועברו בראש המחסנית.

|  |  |
| --- | --- |
| כתובת לחזרה | sp |
| ערך השחקן עבורו רוצים לבדוק האם קיימים מהלכים | 2(sp) |
| ערכים קודמים שהיו במחסנית |  |

מבנה המחסנית בקריאה לפונקציה: