# מגיש:

**שם**: רועי טלבי

**תז**:316411211

# רקע על המשחק

*" שנים רבות הייתה קיימת יריבות בין המלך באווריון (מלך הB) למלך קרימוס (מלך הQ)*

*השניים היו יריבים פוליטיים של אימפריות שונות בחבל הבלקן.*

*כל אחד ניסה לכבוש שטחים רבים יותר מיריבו. אבל האתגר האמיתי היה לכבוש חבל ארץ ארוך יותר מהשני, שניהם נהגו לחסום זה את זה ולקחו חלק במלחמות עקובות מדם..*

*למעשה התחרות בניהם לא הסתיימה מעולם..."*

# מהלך המשחק

למשחק יש שחקנים השחקן האחד זה המחשב והשחקן השני הוא המשתמש.

* המשתמש הוא שחקן שסימנו הוא Q (המלך Q , קרימוס)
* המחשב הוא שחקן שסימנו הוא B (המלך B , *באווריון*)

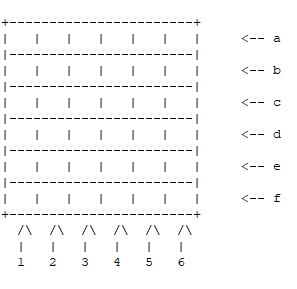
**מטרת המשחק:**  למלא את הלוח ולסיים עם השורה או העמודה הארוכה ביותר , כאשר שורה מוגדרת מרצף סימני השחקן (B או Q)

בכל תור המשתמש בוחר 2 משבצות בלוח שבהם ישבץ את סימני הQ שלו.

כנ"ל לגבי המחשב, הוא בוחר את ה2 משבצות הטובות ביותר לפי דעתו שבהן ישבץ את סימני הB שלו.

המשתמש הוא השחקן בעל התור הראשון לשחק.

המשחק מתחיל עם לוח ריק:



על המשתמש לבחור 2 משבצות **ריקות** בהן לשים את סימנו . מימין ומתחת ללוח יש אינדקסים.

בכל תור, התוכנית תבקש מהמשתמש להכניס קלט מהצורה : X1/N1+X2/N2.

X1,X2 יהיו אותיות בין a לf שייסמנו את השורה הרלוונטית,

N1,N2 יהיו מספרים בין 1 ל 6 שייסמנו את העמודה הרלוונטית.

**לדוגמא:**

המיקום : a/1 יסמן את המשבצת בשורה הראשונה בעמודה הראשונה

המיקום: b/4 יסמן את המשבצת בשורה השנייה ובעמודה הרביעית

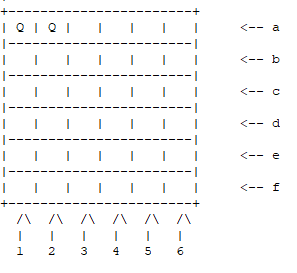
כיוון שנרצה לשבץ 2 סימנים בכל תור, נכניס כקלט 2 מיקומים רצויים בתוספת סימן + באמצע.

**חשוב - לא לשכוח לשים תו נקודה בסוף הקלט !!** ("." הוא תו סיום שורה).

**לדוגמא:**

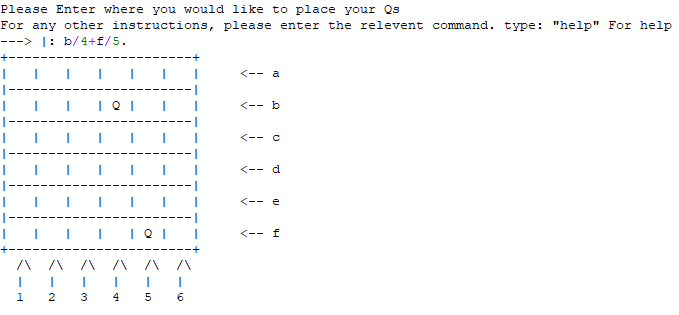
* + אם במהלך הראשון אכניס את הקלט: .**a/1+a/2**

יתקבל הלוח הבא:



* + אם במהלך הראשון אכניס את הקלט: **b/4+f/5.**

יתקבל הלוח הבא:



# פקודות נוספות במשחק

בכל פעם שתתבקש להכניס קלט , קיימים מספר אפשרויות לפקודות/קלטים שתוכל להכניס:

1. help.
   * עבור הפקודה הזאת יוצא מסך עזרה המפרט את הפקודות הקיימות
2. quit.
   * עבור הפקודה הזאת המשחק יסתיים
3. endgame.
   * עבור הפקודה הזאת המשחק יסתיים ויוכרז המנצח בהתאם למצב הנתון כרגע במשחק
4. או קלט תקין מהצורה שהוגדרה לעיל - X1/N1+X2/N2.

**עבור כל קלט אחר תוצג הודעת שגיאה:**



# כיצד להפעיל את המשחק

הקומפיילר הוא LPA prolog . עותק שלו נמצא בתיקיית הפרויקט.

חשוב להפעיל את הקומפיילר דרך קובץ הbat שצורף :START\_PROLOG\_COMPILER.bat

הקובץ יריץ את הקומפיילר עם הגדרות זיכרון גדולות מברירת המחדל. זיכרון זה נדרש לצורך הסריקה באלגוריתם החיפוש.

לאחר הרצת הקומפיילר, יש לפתוח את הקובץ bbq\_ui.pl .

יש לקמפל ע"י Crl+L . ואז להריץ את הפרדיקט start\_game עם דרגת הקושי הרצויה .

כלומר יש להריץ את אחד מהשורות הבאות:

* start\_game(easy).
* start\_game(medium).
* start\_game(hard)

הנחיות יופיעו במהלך המשחק.

# מבנה הפרויקט

קוד הפרויקט יושב תחת תיקיית Code .

התוכנית מחולקת למספר קבצים:

* **bbq\_ui.pl**
  + הקובץ הראשי שאותו יש לטעון לתוכנה, הוא מכיל את כל ממשק המשתמש- הרצת המשחק והדפסת הוראות, הצגת הלוח וכו' ...
  + פרדיקט התחלת המשחק נמצא בקובץ הנ"ל
* **bbq\_gameplay\_hard.pl**
  + הקובץ שמכיל את לוגיקת המשחק בדרגת קושי קשה. (אלגוריתם החיפוש, אפשרויות למהלכים, פונקציה היוריסטית מתאימה לרמת הקושי וכו'..)
* **bbq\_gameplay\_medium.pl**
  + הקובץ שמכיל את לוגיקת המשחק בדרגת קושי בינונית. (אלגוריתם החיפוש, אפשרויות למהלכים, פונקציה היוריסטית מתאימה לרמת הקושי וכו'..)
* **bbq\_gameplay\_easy.pl**
  + הקובץ שמכיל את לוגיקת המשחק בדרגת קושי קלה. (אלגוריתם החיפוש, אפשרויות למהלכים, פונקציה היוריסטית מתאימה לרמת הקושי וכו'..)
* **bbq\_board.pl**
  + מכיל את כל הלוגיקה הרלוונטית ללוח המשחק (השמת סימן במקום בלוח, מציאת טור/שורה הכי ארוכה,..)
* **bbq\_aux.pl**
  + פרידיקטי עזר גנרים ורלוונטים לפרויקט

הקובץ הראשי הוא bbq\_ui.pl הפרדיקט הראשי נמצא שם ושמו הוא start\_game ,

מעבירים את רמת הקושי הרצויה לפרדיקט הנ"ל, הוא דואג בהתאם לרמת הקושי הנתונה לקמפל את קובץ הbbq\_gameplay הרלוונטי .

כמו כן הפרדיקט הנ"ל מקמפל את שאר הקבצים הרלוונטיים לפרויקט.

# מבני נתונים בתוכנית

לוח: **b(Size, Rows)**

כאשר Rows היא רשימה של תווי b/q/n . כל רשימה באורך 6 ויש 6 רשימות.

לדוגמא:

Board = b(6,[[b,b,q,n,n,n],[b,n,n,n,b,n],[b,n,n,n,n,n],[b,n,n,n,n,q],[q,n,n,q,q,q],[n,n,n,n,n,n]])

תו b/q עבור סימן B/ Q בלוח ותו n עבור תא ריק בלוח.

מצב (לטובת אלגוריתם החיפוש): **s(Player, Depth, Board)**

כאשר Board זה הלוח הנוכחי, Player הוא השחקן (player\_b / player\_q)

וDepth זה עומק הסריקה הנתון.

# אלגוריתם החיפוש

אלגוריתם החיפוש הוא אלגוריתם אלפא-בתא שנלמד במסגרת הקורס.

לא ניתן לסרוק את כל המהלכים האפשריים של כל שחקן, זה ייקח זמן רב מדי מכיוון שיש יותר מידי מהלכים אפשריים:

36 משבצות בלוח, 2 אופציות עבור כל סימן . לכן נקבל 2 ^ 36 לוחות אפשריים .

מחשב רגיל לא מסוגל לסרוק את כל האופציות הללו בזמן סביר.

לכן – יצרתי פונקציה היוריסטית מתאימה. וכן נשמור כחלק מהstate בחיפוש את דרגת הסריקה.

ברגע שנגיע לעומק הסריקה הרצוי , הפרדיקט moves לא יחזיר אפשרויות נוספות למהלכים ולכן אלגוריתם אלפא-ביתא יחזיר את התוצאה הטובה ביותר של הפונקציה היוריסטית בסריקה עד העומק המקסימלי שהוגדר.

בהתאם לדרגת הקושי במשחק נבחרה פונקציה היוריסטית מתאימה ודרגת החיפוש הנדרשת נבחרה גם כן בהתאם.

# הפונקציה היוריסטית

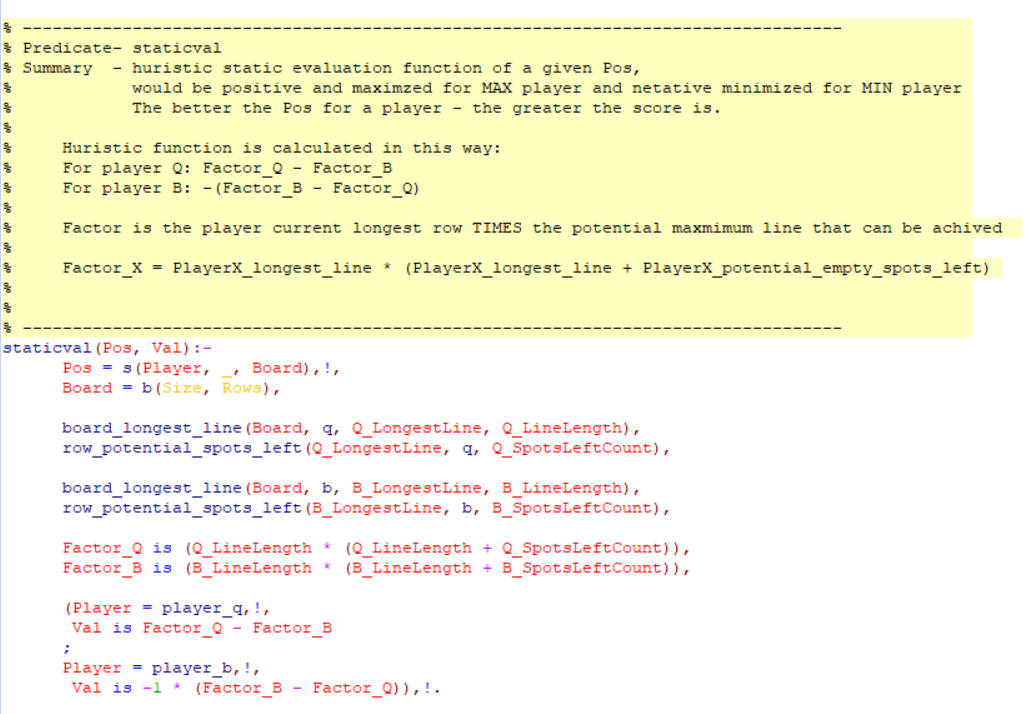
* **דרגת קושי קשה/בינונית:**

כחלק מהפונקציה היוריסטית רציתי להביא לידי ביטוי את מטרת המשחק בצורה הטובה ביותר:

הגדרה – "**קו**" - טור או שורה רצוף של אותו סימן

**המטרה**- להשיג את הקו הכי ארוך (ברצף) של סימני B (אם אתה שחקן B) או סימני Q (אם אתה שחקן Q) .

המטרה שלנו לנצח את שחקן היריב , לכן נרצה שהשורה/הטור הכי ארוך שלנו יהיה ארוך יותר משל השחקן השני. **לכן נרצה תמיד לשאוף ו"לחסום" את השחקן השני** ביחד עם השאיפה שלנו לייצר את השורה או הטור או השורה הכי ארוכה.



נתרגם את זה לעקרונות עבור פונקציה יוריסטית:

* + נרצה שהקו שלנו יהיה ארוך ככל שניתן
  + נרצה שהקו שלנו יהיה ארוך יותר מהקו המקסימלי של היריב
  + נרצה לתת ערך גדול יותר לשורה/טור בו יש מקומות פוטנצאלים נוספים
    - למשל עבור שורה שבסופה יש סימן B (ואני שחקן Q) , האורך המקסימלי של קו שלי יהיה 5 ולא 6 .
    - לכן נכניס בחשבון כמה מקומות נותרים לא חסומים יש בשורה
    - אם השחקן השני כבר "חסם" אותי מלהתקדם באותו קו – נחזיר ערך 0 , לעומת זאת אם יש לי שורה באורך 3 ועוד 3 מקומות פנויים – מס' המקומות הפוטנציאלים יהיה 3 .

לצורך כך , נגדיר ערך FACTOR , עבור כל שחקן ולוח משחקן נתון:

Factor :

Factor יהיה: הקו הכי ארוך \* (הקו הארוך + מספר המקומות הפוטנציאלים שעוד נותרו ויהיה אפשר למלא אותם) .

כלומר במילים: הקו הארוך ביותר הנוכחי כפול האורך הפוטנציאלי המקסימלי שלו

Factor\_X = PlayerX\_longest\_line \* (PlayerX\_longest\_line + PlayerX\_potential\_empty\_spots\_left)

ערך הפונקציה היוריסטית עבור שחקן MAX יהיה

Val = Factor\_X – Factor\_Y

ועבור שחקן MIN:

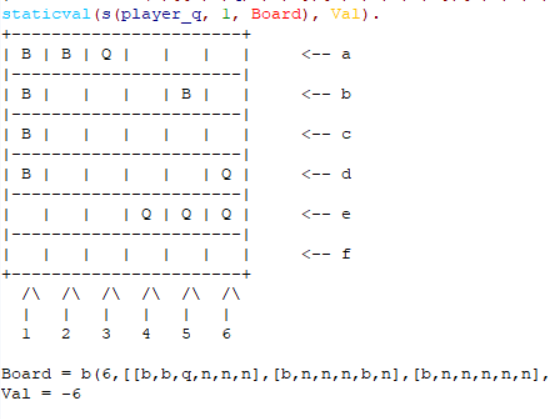
Val = -( Factor\_X – Factor\_Y)

מהפונקציה היוריסטית הזאת נקבל כמה יתרונות:

* + עודדנו את שחקן המחשב "לחסום" את קווי היריב, בכך שאם יחסום אותו – מספר המקומות הפוטנציאלים של שחקן היריב יהיה 0 וכך ערך הFACTOR שלו יהיה קטן יותר וההערכה היוריסטית טובה יותר.
  + שחקן המחשב ישאף להמשיך את הקו שלו שיהיה ארוך ככל שניתן , שכן היא חלק מחישוב הFACTOR עבור אותו שחקן

דוגמא:

במצב הנ"ל שחקן Q (שחקן MAX) נדרש לשבץ סימן Q נוסף. ההערכה היוריסטית במצב הנוכחי היא לטובת שחקן B שכן יש לו שורה ארוכה יותר .



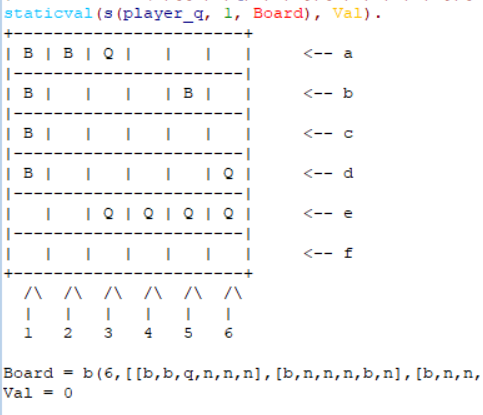
ההערכה תהיה :

Factor\_Q = 3 \* 6 = 18

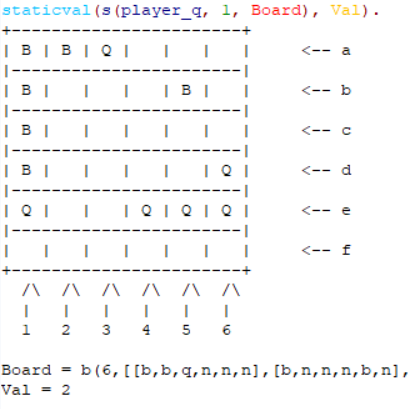
FactorB = 4 \* 6 = 24

Val = 18-24 = -6

* אם Q יבחר להתקדם ולשים עוד סימן Q כדי להאריך את השורה שלו נקבל הערכת Val של 0 .

****

* לעומת זאת אם Q יבחר "לחסום את שחקן B" נקבל הערכה טובה יותר:

****

Factor\_Q = 3 \* (3+3) = 18

Factor\_B = 4 \* (4+0) = 16

Val = Factor\_Q - Factor\_B = 2.

**הפונקציה היוריסטית בדרגת קושי בינונית, קשה היא אותה פונקציה. ההבדל**

**הוא בעומק הסריקה באלגוריתם אלפא-בתא.**

**בדרגת קושי קשה נסרוק עד 3 רמות ואילו בדרגת קושי בינונית נסרוק עד 2 רמות.**

* **דרגת קושי קלה:**

כאן נרצה להקל על השחקן על מנת שיהיה לו קל לנצח.

עם זאת לא נרצה ששחקן "המחשב" יהיה טיפש לגמרי וישחק מהלכים רנדומליים.

לכן המוטיבציה לפונקציה יוריסטית תהיה:

* המחשב ירצה ליצור קו ארוך ביותר
* לא נרצה שהמחשב יחסום את השחקן השני ויקשה עליו מידי.

לכן הפונקציה היוריסטית תהיה:

Val = Player\_X\_longest\_line - Player\_Y\_longest\_line

(כמובן שבשחקן MAX יהיה חיובי ובשחקן MIN יהיה שלילי)