**מסמך תיעוד חיצוני**

**מבני נתונים מרכזי:**

רשימת הקלפים – רשימת הקלפים בנויה משלוש שדות Card-Jokers-Num :

Card- רשימה המכילה את הקלפים שאינם ג'וקר ממוינת בסדר שאינו יורד.

עבור השחקן האנושי בלבד Card יכולה להיות רשימה של 4 תתי רשימה Reds,Green,Yellow,Blue שבה כל רשימה מכילה את כל הקלפים בצבע המתאים ממוינים בסדר שאינו יורד.

Jokers- רשימת ג'וקרים.

Num – ערך מספרי המציין את מספר הקלפים (כולל ג'וקרים).

* + (הערה חשובה - אנו לא מעדכנים את מספר הקלפים בסיום כל תור כך שלרוב Num=11 ).

alpha(NumA, Lonely,Order)- פרדיקט הנטען למערכת השומר את רשימת הקלפים שלא נמצא עבורם סידור חוקי, Lonely, את מספרם, NumA,ואת הסידור שנמצא עד כה.

* + (הערה חשובה – גם את NumA אנו לא מעדכנים לאחר הורדת קלף).

**מימוש הבדיקות - עץ המשחק:**

תחילה נדאג שהקלפים של המחשב מסודרים תמיד בסדר שאינו יורד ללא חשיבות לצבע (ישמש אותנו לצמצום הבדיקות) ונגדיר עץ באופן הבא: לכל צומת בעץ יש חמישה שדות, סידור חוקי שנמצא עד כה המכיל (לכל הפחות) את הקלף הנוכחי, הקלפים שעדין לא נבדקו, temp , הקלפים שבחרנו להגדיר אותם כבודדים, והקלפים שנמצא עבורם סידור חוקי מסודרים ברשימות (רשימה לכל סידור). לרשימת הקלפים שנמצאים בסידור חוקי, רשימת הקלפים שנשאר לסדר, temp, ורשימת הקלפים הבודדים נוסיף שדה השומר את מספר הקלפים ברשימה.

play\_tree(In\_progress-Num1,Rest\_cards-Num2,Temp\_cards-Num3,Lonely\_cards-Num4,Order)

מכל צומת בעץ יוצאת מספר קשתות כמספר הסידורים האפשריים המתחלים בקלף הראשון ב Rest. כל צומת מטופלת באופן הבא:

1. אם הסדרה שנמצאה עד כה היא חוקית (באורך 3 או יותר) היא נשמרת ברשימת ב Order כרשימה.
2. אם הסדרה היא באורך שתים היא נשמרת בשדה הבודדים כרשימה, לרשימת הבודדים נוסיף את הערך 1.5. אנו מחשבים כל קלף מהזוג כ 0.75 באופן הזה ניתן עדיפות לסידורים המכילים יותר זוגות לעומת סידורים שאינם מכילים זוגות.
3. אם זה איבר בודד אנו בודקים:
4. האם קיים איבר שערכו הוא X+2 והוא בעל אותו צבע (למשל, עבור הקלף שערכו 1 נחפש את הקלף מאותו צבע שערכו 3) אם איבר זה קיים נכניס את האיבר [X,X+2] לרשימת ה temp נוסיף למספור של רשימת temp אחד.
5. האם קיים איבר ברשימת temp מהצורה [X-2,X] , נוציא אותו מהרשימה ונוסיף את הרשימה [X-2,X] לרשימת הבודדים. ונוסיף 1.5 למספור רשימת הקלפים הבודדים.

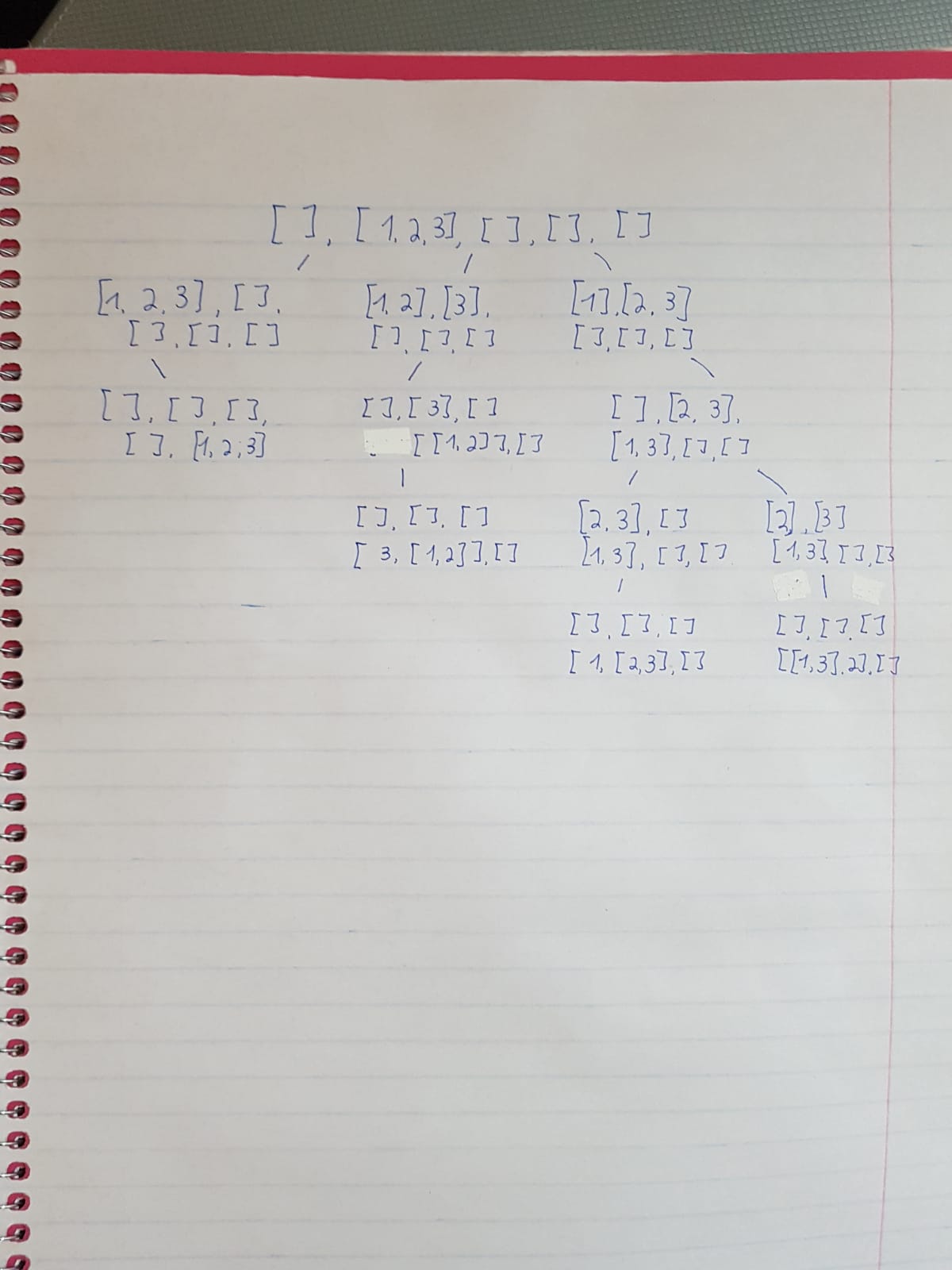
אם שני האפשרויות אינן מתקיימות, נכניס אותו לשדה של הבודדים.

בסיום הבדיקות נמשיך למציאת סידורים אפשריים נוספים מרשימת הקלפים שנותרה. עד שלא נותרו קלפים, אז נגיע לעלה.

עלה יוגדר להיות []-0,[]-[]-0,Temp-Num1,NewLonely Num2,Order)).

בתחילת הטיפול בעלה נעביר את כל הקלפים שהם הקלף הראשון של זוג הנמצא ב temp לרשימת הבודדים. NewLonely) הוא Lonely המכיל בנוסף את כל ה X מהצורה [X,\_] הנמצאים בtemp).

כמו כן נשמור את רשימת הקלפים הבודדים בפרדיקט save. ואם מספר הקלפים הבודדים קטן ממספר הקלפים הבודדים שנמצא עד כה אנו מעדכנים את אלפא. עץ המשחק יכול להצליח אם"ם בסיום הבדיקות מתקבל עלה שבו מספר הקלפים הבודדים הוא 0 או 1 כאשר הרמה היא 1. בכל מקרה אחר עץ הבדיקה נכשל.

*מצורף איור לדוגמא עבור רשימת הקלפים* [1,2,3]

עץ המשחק מנוהל על ידי play\_tree(List) המפעיל את play\_tree([]-0,List,[]-0,[]-0,[]) ,List רשימת הקלפים. בשונה מעץ המשחק play\_tree(List) *תמיד מצליח.*

*צמצום סריקת העץ* – פונקציה היוריסטית*:*

בדי לצמצם את עץ החיפוש נבטיח שהבחירה הראשונה של הקשת השמאלית מבין הקשתות השמאליות היא בעלת סידור הקלפים המקסימלי מבין הסידורים האפשריים. באופן הזה בכניסה הראשונה לעלה הסידור שנמצא הוא (באופן נאיבי) הטוב ביותר. נשתמש בפרדיקט alpha(α, Lonely,Order) השמור במערכת:

* אם הצומת הנוכחית היא עלה וגם α |Lonely|< עדכן את α להיות |Lonely| ואת Lonely,Order בהתאם.
* אם הצומת הנוכחית היא קשת לפני הכניסה לצומת נבדוק את הפונקציה היוריסטית:

1. אם α=1וגם הרמה היא 1 או α=0 הפונקציה נכשלת.
2. אחרת נחשב את הערך:

אם αf≤ נכנס לצומת -הפונקציה מצליחה. אחרת, לא נכנס לצומת.

הפונקציה מחשבת את מספר הקלפים הבודדים שיש כרגע, מחושב בזמן קבוע. אם עד כה צברנו יותר קלפים בודדים ממספר הקלפים הבודדים שנמצאו עד כה אנו לא ממשיכים בחיפוש בכיוון הזה. אם מספר הקלפים הבודדים שנמצאו הוא 1 או 0 אנו לא בוחנים אפשרויות נוספות.

עץ המשחק נעזר בפרדיקטים הבאים:

color(Card,List ,New\_rest,In\_progress)

series(Card,List,New\_rest,In\_progress)

Card- הקלף ממנו תתחיל הבדיקה,List -היא רשימת הקלפים מהם ניתן לבחור סידור בנויה מ Rest-- Rest -Jokers-Left\_num, Left\_num- מספר הקלפים הנמצאים ב Rest, -New\_rest רשימת הקלפים שנותרה לאחר הסידור בנויה מ Cards-Jokers-Num , In\_progress -בנויה מ Order-Num, Order רשימת קלפים מתוך Rest הכולל גם את Card המכילה סדר חוקי בין Num קלפים.

שתי הפרדיקטים מאפשרים קריאה מחדש (redo) והם בונים סידור מ

1. קלפים בעלי אותו צבע בסדרה חשבונית כאשר d=1.
2. קלפים בעלי אותו ערך מספרי בצבעים שונים.

שתי הפרדיקטים נותנים עדיפות דומה לבחירות הסידור שהם מחזרים, סדרי העדיפויות הם:

* 1. סידור באורך מירבי שאינו כולל ג'וקרים.
  2. סידור בארוך מירבי הכולל ג'וקר אחד.

.

\_.1. סידור בארוך מירבי הכולל את כל ג'וקרים.

2.1. סידור באורך כמעט מירבי הנבנה מהסידור הראשון ללא הקלף האחרון ללא ג'וקרים.

2.2. סידור באורך כמעט מירבי הנבנה מהסידור הראשון ללא הקלף האחרון עם ג'וקר אחד.

.

\_.2. סידור באורך כמעט מירבי הנבנה מהסידור הראשון ללא הקלף האחרון עם כל הג'וקרים.

.

.

בשונה מ color series אינו יכול להחזיר סידור באורך אחד.

Color ממומש על יד color\5 המכיל בנוסף למשתנים של color\4 רשימה של הצבעים הנמצאים בסידור שנמצא עד כה. color\4 מפעיל את color\5 או את add\_jokers, המאפשר יציאה והחזרת קלף בודד.

קלף נוסף נכנס לרשימה אם ורק אם הוא בעל צבע שאינו מופיע ברשימת הצבעים, לאחר הוספת קלף חדש לרשימת הקלפים ניתן להמשיך לצורך חיפוש קלפים נוסיפם או לצאת ל add\_jokers. כמו כן קיימת כניסה נוספת המדלגת על הקלף הנוכחי וממשיכה לקלף הבא, כניסה זאת אפשרית רק אם הקלף עליו אנו מדלגים הוא בעל אותו ערך כמו הקלף המתחיל את הסידור, מכניסה זאת לא ניתן לצאת ל add\_jokers (כך שהאפשרות היחידה לסיים סידור היא על ידי הוספת קלף נוסף לסידור). Color נכשל כאשר הקלף הבא ב Rest אינו בעל אותו ערך מספרי כמו הקלף שאיתו אנו התחלנו את הסידור.

add\_jokers מאפשר ל color להצליח והוא בעל שתי כניסות הכניסה הראשונה מחזירה בתוצאה את הערכים שהתקבלו עד כה (הסידור ורשימת הקלפים החדשה) ואילו הכניסה השנייה מוספיה ג'וקר אחד לרשימה שהתקבלה עד כה ומפעילה שוב את add\_jokers.

Series מורכב מ series2 ומ series1 והוא בעל שני כניסות לכל אחד מהם.

series1 - הכניסה הראשונה מתאפשרת אם ורק אם, הקלף הראשון ב Rest (Card2) הוא קלף בעל צבע זהה לצבע של הקלף Card1 -הקלף שהתקבל כקלט ל series1 וערכו הוא ערך הקלף פלוס אחד. במקרה זה הקלף נכנס לסידור הנבנה. קיימת שלושה אפשרויות (בסדר הזה) מהם ניתן לבחור את ההמשך בכניסה זו:

1. להמשיך ל series1 למציאת קלף נוסף הממשיך את הסדרה, אנו שולחים את הקלף Card2 כקלף הקלט (series1(card(Value2,Color),List,New\_list,Series-Num))
2. לסיים בעזרת done המחזיר בפלט את הערכים שנצברו עד כה.
3. להמשיך ל series2.

הכניסה הבאה של series1 מאפשרת לדלג על הקלף הנמצא בראש הרשימה וממשיכה ל series1 שוב. (שוב, לא ניתן לצאת מכניסה זאת).

series2 הוא בעל כניסה אחת הדומה לכניסה הראשונה של series1 אך בשונה ממנה הקלף אותו אנו מוספים לרשימה הוא ג'וקר. היא גם בעלת שלושה אפשרויות להמשך:

1. להמשיך ל series1 למציאת קלף הממשיך את הסדרה, אנו שולחים את הקלף שאותו מחליף הג'וקר כקלף הקלט (series1(card(Value+1,Color),List,New\_list,Series-Num)).
2. לסיים בעזרת done המחזיר בפלט את הערכים שנצברו עד כה.
3. להמשיך ל series2, להוספת ג'וקר נוסף.

has\_card(Card,List-Jokers-Num,Temp,New\_temp)

מקבל קלף (Card) ובודק האם ב List, רשימת הקלפים שנותרו, יש קלף שערכו הוא בערך הקלף פלוס שתים וצבעו כצבע הקלף. אם כן אנו מכניסים ל Temp את זוג הקלפים כרשימה [Card1,Card2] ומחזירים אותה הרשימה temp לאחר ההוספה ב New\_temp. ל has\_card יש שני כניסות אחת לביצוע השינוי כאשר הקלף Card2 נמצא בראש רשימת הקלפים והשנייה לדילוג על הקלף הראשון ברשימה וחיפוש בהמשך הרשימה. הכניסה השנייה מתאפשרת רק כאשר הערך של הקלף בראש הרשימה קטן מערך הקלף עבורו מתבצע החיפוש פלוס 2. כך שכאשר הגענו לקלף שערכו גדול ביותר מ 2 מהקלף עבורו מתבצע החיפוש has\_card נכשל.

has\_temp(Card,Temp,Lonely,NewTemp,New\_Lonely).

מקבל קלף ובודק האם הוא חלק מזוג ברשימה Temp. אם כן הזוג יוצא מרשימת Temp, המוחזרת ב NewTemp , ונכנס לרשימה Lonlely המוחזרת ב New\_Lonely. ל has\_temp יש שני כניסות כניסה ראשונה מתאפשרת רק אם בראש רשימת הקלפים Temp קיים הזוג card(Value2,Color),card(Value1,Color)]] , כאשר card(Value1,Color) הוא הקלף עבורו מתבצע החיפוש- קיום זוג כזה ברשימת Temp אפשרי רק אם Value2+2= Value1. הכניסה השנייה מתאפשרת רק כאשר הקלף הראשון מבין זוג הקלפים הנמצא בראש רשימת Temp גדול מהערך של הקלף עבורו מתבצע החיפוש . הרשימות ב Temp מוכנסת כך שהזוג הראשון ברשימה הוא הזוג עם הערכים הגדולים יותר, כך שאם הגענו לזוג שערך האיבר הראשון שלו קטן יותר מהערך של הקלף לא יתכן שיש בהמשך זוג העונה לדרישות שלנו.

***מהלך המשחק*:**

המשחק מתחיל בחולקה אקראית של 10 קלפים לכל שחקן. עם קבלת הקלפים השחקן האוטומטי ממיין את קלפיו בסדר עולה מכניס את alpha(∞,\_,\_) .בכדי להבטיח עדכון של alpha ומפעיל בפעם הראשונה את עץ המשחק. בסיום ההפעלה alpha עודכנה. אם 0=α המחשב ניצח. הקלפים של השחקן האנושי ממוינים תחליה בסדר עולה ולאחר מכאן לפי צבע, באופן הזה אנו מבטיחים שבסידור על פי צבע כל הקלפים מאותו הצבע יהיו ממוינים בסדר עולה. וכן שכאשר הקלפים הם בסדר עולה יהיו תת סידור לפי צבע האדומים תחילה, ירוקים, צהובים, כחולים.

בתחילת המשחק המחשב עורך הגרלה הקובעת מי מתחיל.

תור המחשב:

מכאן והלך השחקן האוטומטי בודק, האם ניתן להגיע לערך טוב יותר, α נמוך יותר, בעזרת לקיחת הקלף שהיריב הוריד, אם כן הערך של alpha(α, Lonely,Order) מתעדכן בהתאם. הבדיקה הזאת לא מתבצעת כאשר המחשב מתחיל וזהו תורו הראשון. הדבר מתבצע באופן הבא: השחקן מכניס את הקלף שהוריד השחקן השני למקומו המתאים ברשימת הקלפים, שומר את ערך α הנוכחי ב α1 ומפעיל את עץ הסריקה, אם יש שינוי בערכו של α כלומר α1≠α2 כאשר α2  הוא ערכו של α כעת, המחשב מכריז כי הוא לקח את הקלף אותו הוריד היריב וממשיך משם להורדת הקלף המתאים. אחרת, המחשב לוקח קלף חדש מהקופה וממכניס אותו למקמו המתאים ברשימה, מפעיל את עץ הסריקה ומוריד את הקלף הרלוונטי, בסיום ההורדה רשימה הקלפים הבודדים ביחס alpha מתעדכנת וכן רשימת הקלפים של השחקן.

כאשר α=1 או α=0 המחשב מכריז כי הוא ניצח, אחרת המחשב מסיים את תורו ומעביר את התור לשחקן.

תורו של המחשב מנוהל על ידי computer\_turn(ListPlayer,ListComputer,X) (X- הקלף אותו ניתן להרים. ו checking(ListPlayer,ListComputer,Num2).

computer\_turn הוא בעל שני כניסות הכניסה הראשונה מטפלת בלקיחת הקלף אותו הוריד השחקן בתור הקודם (X) היא מצליחה אם"ם α1≠α2 (כאשר כך הם פני הדברים ישנו cut המונע כניסה לאפשרות השנייה) בכניסה השנייה השחקן לוקח קלף חדש. במהלך כל הפעלה של computer\_turn מוכנס הקלף הרלוונטי לרשימת הקלפים, ומופעל עץ הסריקה.

במהלך ההפעלה של computer\_turn בכניסה השנייה (בלבד) מתבצע עדכון alpha כך שרשימת הקלפים תכיל גם את הקלף החדש שהתקבל. העדכון מתבצע בכדי לאפשר בחירת הקלף החדש כקלף להורדה. הקלף מוכנס כקלף הראשון ברשימת הקלפים הבודדים, ובמקרה של העדר שינוי ב alpha הוא יבחר כקלף להורדה.

checking – מופעל בסיום computer\_turn זהות הכניסה שהפעילה את checking נשלחת ב Num2 (לצורך הממשק בלבד) checking מבצע את הבדיקות הבאות:

1. התאמת רמה **adapter\_level** מסתיים כאשר מספר הקלפים הבודדים ב alpha גדול מאחד או כאשר הרמה שווה לאחד.
2. בדיקה האם השחקן יכול לנצח ללא הורדת קלף (נועד למנוע כניסה ל drop עם רשימת קלפים ריקה).
   1. אם כן, סיים וקרא ל win\_the\_gamae (חלק מהממשק הגרפי).
   2. אחרת,
      1. בחירת קלף להורדה (drop) ומחיקת הקלף שהורד מרשימת הקלפים של המחשב וגם מרשימת הקלפים הבודדים של alpha (על ידי עדכון).
      2. בצע הדפסות (לקחתי קלף חדש/ לקחתי את הקלף שהורדת והורדתי …).
      3. בדיקה האם המחשב ניצח בסיום הורדת הקלף.
         1. אם כן, סיים וקרא ל win\_the\_gamae.
         2. אחרת, העבר את התור לשחקן.

תור השחקן:

תור השחקן מנוהל על ידי follow רשימת פרדיקטים בעלת מבנה של עץ הבנויה מ 6 שדות, בסיום חלק מהפרדיקטים השחקן מתבקש לענות על שאלה, תשובתו לשאלה קובעת את ההמשך. השחקן יכול לנוע בין הכניסות השונות בהתאם לקלט שהוא מכניס.

follow(Command,Direction,X,ListPlayer,NewListPlayer,New\_X)

Command- הבחירה של השחקן, כאשר לשחקן אין אפשרות לבחור nil השחקן יכול תמיד לבחור exit לצורך יציאה, help לקבל עזרה, aaa/abc לשינוי סדר הקלפים.

Direction- מגדיר מהו הפרדיקט הבא.

X – משתנה, בהתחלה זהו הקלף שהמחשב הוריד קודם, אח"כ הקלף אותו בחר השחקן להכניס.

ListPlayer- רשימת הקלפים של השחקן.

- NewListPlayer,New\_Xהתוצאה, רשימת הקלפים של השחקן והקלף אותו הוא בחר להוריד.

Follow מאפשר לשחקן לבקש עזרה וכלל מערכת לאיתור ותיקון שגיאות.

בסיום תורו של השחקן המחשב בודק האם השחקן ניצח, אם כן ההרצה מסתיימת אחרת התור עובר למחשב. בדיקה מבוצעת על ידי player\_win(List) המקבל את רשימת קלפיו של השחקן ומצליח אם"ם ניתן לסדר את כל קלפיו של השחקן בסדר חוקי. הבדיקה מבוצעת על רשימה ממוינת בסדר לא יורד בלבד (לפני הבדיקה מתבצעת התאמה) והיא נעזרת ב series ו- color.

**רמות:**

המשחק כולל 4 רמות. מימוש הרמות מתבצע על ידי שיפור הבחירות אותם מבצע המחשב- בחירת קלף להורדה, הרמת הקלף אותו הוריד המחשב או לקיחת קלף חדש. בכדי לבצע בחירות אלו בצורה המיטבית על המחשב להתחשב בכל הסידורים האפשריים. בכדי להגדיל את מספר הסידורים האפשריים נוכל להוסיף לשחקן ג'וקר אותו הוא יהיה חייב להוריד לפני שהוא מכריז על ניצחון, באופן הזה אנו מתחשבים תחילה בכל הסידורים האפשריים עם הנחה שבכדי להשלים את הסידור צריך שיצא הקלף המתאים ורק לקראת הסוף אנו נמנעים מהנחה זאת. למשל עבור הקלפים 4 5 4 5 7 8 יהיה זה נכון יותר להוריד את אחד הקלפים האדומים ולא את אחד הכחולים. עץ המשחק אומנם מזהה את כל הרצפים האפשריים אך אינו מתחשב בכך ש 6 משלים את שתי הרצפים הכחולים ולכן לא תתגלה עדיפות בעת בחירת הקלף להורדה בין הכחולים לאדומים, מה שבסדר עבור הרמה הקלה. על ידי הוספת ג'וקר נוכל לפתור בעיה זו ועץ המשחק כן יתחשב ברצף המלא של הכחולים. באופן דומה עבור רשימת הקלפים 11 11 11 7 9 עדיף להוריד את הקלף 7 ולא את 9 מתוך הנחה שאולי יצא 11 ו- 10 עץ המשחק אינו מבחין בכך (ואין צורך להתחשב בכך לקראת סיום המשחק) הוספת ג'וקר פותרת את הבעיה. נכון לציין שלמרות שההחלטה היא מושכלת יותר אין זה בהכרח שהיא הטובה יותר היות והקלף הבא שיתקבל תלוי במזל. למשל בדוגמה הראשונה יכולים להתקבל 3 ו- 6 מה שיהפוך בדיעבד את ההחלטה להוריד את 7 8 להחלטה הנכונה יותר.

**מתחילים:**

רשימת הקלפים נותרת ללא שינוי.

**בינוניים:**

מוסיפים ג'וקר אחד למחשב.

**מתקדמים:**

מוסיפים שני ג'וקרים למחשב.

**מומחה:**

מוסיפים שלושה ג'וקרים למחשב

פרדיקטים מממשים:

adapter\_level(Cards-Jokers-Num,Result) -

כאמור, המחשב אינו יכול לנצח עם הג'וקרים. המחשב יכול להשתמש בג'וקרים עד שהוא מגיע לסידור מלא של הקלפים כזה שלא משאיר אף קלף בודד אחד. מספר הקלפים הבודדים לאחר הורדת הג'וקר הוא לכל היותר 4, כפי שמתקבל במקרה של הדוגמה הראשונה, וערכו הוא לכל היותר 3 (2 רצפים). הפרדיקט adapter\_level(Cards-Jokers-Num,Result) מתאים את הרמה בהתאם למספר הקלפים הבודדים. כאשר מספר הקלפים הבודדים הוא 0 הרמה יורדת באחד עד אשר הרמה היא 1 או שמספר הקלפים הבודדים גדול מ 0. בסיום כל הורדת רמה המחשב מפעיל שוב את עץ המשחק עם alpha(4,nil,nil). Result – היא הרשימה של המחשב לאחר מחיקת מספר הג'וקרים הנכון. adapter\_level/2 עורך גם את השינויים הנדרשים בפרדיקטים level ו- save.

level\_config(List,Result) -

קביעת הרמה מתבצעת בתחילת המשחק על ידי level\_config(List,Result) המאפשר לשחקן לבחור את הרמה. level\_config מקבל את List רשימת הקלפים של המחשב ומחזיר את רשימת הקלפים לאחר השינוי הנדרש. בחירת השחקן מעוברת ל adapter\_level/3 המבצע את התאמות ואת הכנסת הרמה המתאימה וכלל מערכת לתיקון שגיאות. בסיום מופעל עץ המשחק (ההפעלה הראשונה שלו) ואנו בודקים שוב את ההתאמת הרמה על ידי adapter\_level/2.

**הכנסת קלף**:

הכנסת קלף של המחשב ושל השחקן כאשר רשימת קלפיו היא בסדר שאינו יורד מתבצעת על ידי insert\_card(Card,Cards-Jokers-Num,New\_cards- New\_Jokers- New\_Num) . Card- הקלף להכנסה.

אם הקלף להכנסה הוא ג'וקר הוא נכנס לרשימת הג'וקרים, בכל מקרה אחר מופעל insert\_card(Card,Cards,New\_cards) המזהה את מיקומו של הקלף על ידי חיפוש הקלף הראשון שערכו גדול או שווה לערך הקלף להכנסה ומכניס אותו לפניו.

השחקן מפעיל את insert\_card על ידי insert הבודקה תחילה האם ניתן לפרק את רשימת הקלפים לארבע רשימות קלפים, אם כן אז רשימת הקלפים של השחקן ממוינת על פי צבע במקרה זה נזהה את הצבע של הקלף להכנסה ונכניס את הקלף למקומו המתאים בעזרת insert\_card אם רשימת הקלפים מהצבע המתאים. אם לא ניתן לפרק את רשימת הקלפים לארבע רשימות insert מפעיל את insert\_card על כל רשימת הקלפים.

## בחירת קלף להורדה

### בחירת קלף להורדה - המחשב

בכל הגעה לעלה המחשב שומר את רשימת הקלפים הבודדים שנמצאה ולפני כל כניסה לתורו המחשב מסיר את הזיכרונות הקודמים. כך שרשימת הקלפים הבודדים השמורה היא זאת שהתקבלה במהלך התור עצמו. בחירת הקלף להורדה מתחלקת לשניים כאשר יש זיכרונות וכאשר אין זיכרונות. כאשר יש זיכרונות אנו סופרים עבור כל קלף ברשימת הבודדים הנמצאת ב alpha כמה פעמים הוא מופיע בתוך הרשימות הנמצאות בזיכרון. הקלף היורד הוא הקלף אשר מספר מופעיו הוא הגדול ביותר ואינו חלק מזוג, במידה וברשימת הבודדים השמורה ב alpha אין קלפים בודדים, הקלף הנבחר להורדה הוא הקלף בעל מספר המופעים הגדול ביותר מבין הזוגות. במקרה זה עלינו גם לעדכן את הערך של מספר הקלפים השמור ב alpha. הערך שמקבלים זוג קלפים הוא 0.75 לכל קלף, פירוק זוג ולקיחת קלף חדש משאירה אותנו עם שני קלפים בודדים שערכם הוא 2.

* למשל עבור הסדרה הבאה [2,2,3,4] התוכנת תבחר להוריד 4 היות ומספר הקלפים הבודדים הקטן ביותר שניתן להגיע עליו הוא 2 ושני הסידורים האפשריים הם 2,2,\_ ואז הקלפים הבודדים הם 3,4 או 2 ,3,\_ ואז הקלפים הבודדים ,2 ,4 מספר מופעיו של 4 הוא הגדול ביותר ולכן הוא זה שירד.

האפשרות השנייה היא שאין זיכרונות במקרה הזה המחשב יוריד את הקלף הראשון מתוך רשימת הבודדים ב alpha שאינו חלק מזוג.

בעת בחירת קלף להורדה אנו נבדוק שהקלף הנבחר אינו ג'וקר.

הפרדיקטים המטפלים בבחירת הקלף להורדה הם:

drop(Lonely,Card)

Lonely- רשימת הקלפים הבודדים ביחס alpha . Card- הקלף הנבחר להורדה בסיום התהליך.

הפרדיקט הוא בעל שני אפשרויות כניסה כאשר בראשונה יש cut אדום המונע כניסה לפרדיקט הבא בסיום הבדיקה.

הפרדיקט drop נעזר ברשימת הפרדיקטים הבאה:

1. counter(List,Saves,Resule)

* סופרת את מספר המופעים של כל איבר ב List ב Saves.

List - רשימת הקלפים הבודדים השמורה ב alpha.

Saves -רשימה המכילה רשימות בל אחת מהם היא אחד מהרשימות הקלפים הבודדים שנשמרו

Resule - רשימה הזהה ל List אך כל איבר בה הוא מהצורה Card-Nunber\_of\_shows.

1. count(Card,Saves,Num) – מבצע את הספירה עצמה. מקבל את הקלף עבורו תתבצע הספירה ואת רשימת הרשימות ב save ומחזיר את מספר המופעים של הקלף.
2. max(Shows,Result)

* מחזירה את הקלף בעל מספר המופעים הגדול ביותר מתוך Shows ב Result.
* בעלת שתי כניסות עם cut אדום אם הכניסה הראשונה הצליחה הראשונה.
* הכניסה הראשונה מחזיר את הקלף שמספר מופעיו הוא מקסימלי ואינו חלק מזוג הכניסה מחזירה את הקל שמספר מופעיו הוא מקסימלי והוא חלק מזוג ואינו ג'וקר.

בחירת קלף להורדה - השחקן

השחקן בוחר את הקלף שברצונו להורידה על פי ציון מקומו או על ידי הכנסת מספר הקלף וצבעו.

ציון המקום נעשה בעזרת printNumbering המקבל את רשימת הקלפים ומדפיס את האינדקס של כל אחד מהם במקומו המתאים.

בבחירה על ידי מיקום הורדת הקלף מתבצעת על ידי:

drop\_player(List,Num,Newlist,Card) ו- drop\_player(List,Num,Return\_num,Newlist,Card)

List – רשימת הקלפים של השחקן מכילה קלפים בלבד (בעלת שתי , Num- האינדקס של הקלף להורדה, Newlist – רשימת הקלפים בסיום ההורדה, ו – Card הקלף אותו בחר השחקן להוריד. Return\_num – מחזיר את המספר שנשאר לבדוק בכדי להגיע לקלף עם האינדקס הנכון.

ל- drop\_player/4 יש שני אפשרות כניסה עם cut אדום בכניסה הראשון. הכניסה הראשונה מתאפשרת כאשר האיבר הראשון ב List הוא קלף מה שמעיד שהרשימה ממוינת בסדר שאינו יורד, הכניסה השנייה מתאימה לרשימת קלפים ממוינת על פי צבע (רשימת רשימות). בכניסה הראשונה אנו בוחרים את הקלף להורדה בעזרת drop\_player/5 וכאשר Card==nil - עדין לא נבחר קלף להורדה, אנו מורידים ג'וקר. בכניסה השנייה נעשה גם שימוש ב drop\_player/5 אך הפעם אם בסיום לא מתקיים Card\=nil מתבצעת קריאת drop\_player/4 פעם נוספת אך הפעם האינדקס Return\_num. הכניסה השלישית מיודעת להורדת ג'וקר כאשר הרשימה ממוינת על פי צבע ולא נשארו איברים ברשימה (רשימה ריקה)

בעת בחירת קלף להורדה על ידי הכנסת ערך הקלף וצבעו אנו משתמשים ב delete\_card\_player אשר בדומה להכנסת הקלף של השחקן מזהה את הסידור של קלפי השחקן. אם הסידור הוא בסדר שאינו יורד אנו מפעילים את delete\_card אחרת אנו מזהים את הצבע של הקל להורדה ומפעילים delete\_card על תת הרשימה המכילה את כל הקלפים מאותו צבע.

**הדפסת קלפים:**

printCards( Cards-Jokers-Num) מפעיל את printCards(Cards) ואת printCards(Jokers).

הדפסת הקלפים מבוצעת על ידי printCards(List) המטפלת שני מקרים: הדפסת הקלפים של רשימה והדפסת קלפים של רשימת רשימות. הדפסת קלפים של רשימת רשימות מבוצעת על ידי קריאה להדפסת קלפים של רשימה לכל אחת מתתי הרשימות.

printCards1, printCards2, printCards3 מטפלת בהדפסת קלפים של רשימה משולבת, כאשר תתי הרשימות מודפסים תחילה. printCards2 מדפיס רק את תתי הרשימות ברשימת המקור, ו printCards3 מדפיס את האיברים שאינם רשימה עצמאית.

printNumbering- פועל באופן אנלוגי ל printCards אך בשונה ממנו הוא קיים גם כ printNumbering/1 וגם כ printNumbering/3. printNumbering/1 מפעיל את printNumbering/3 פעמיים, פעם אחת על רשימת הקלפים ופעם שניה על רשימת הג'וקרים. printNumbering(List,Num,Next) מקבל את List כרשימת הקלפים על פיה יש להדפיס. הרשימה משמשת גם לצורך קביעת מספר האינדקסים, אך גם (ובעיקר) לצורך קביעת מיקום האינדקס ביחס לשאר (למשל עבור 10 ועבור 1 הדפסת האינדקס באותו מקום תגרור הסטה של האינדקסים הבאים). Num- הוא האינדקס, Next הוא האינדקס של האיבר המגיע בסיום הדפסת הרשימה List.

**יצירת הקלפים וקבלת קלף חדש:**

יצרת הקלפים מתבצעת על ידי create\_card המפעיל את create\_card(Value,Color, Serial\_num)

Value- ערך הקלף, Color- צבע הקלף, Serial\_num- המספר הסידורים של הקלף,

עבור (1,red,1) המכניס את כל הקלפים האדומים כפרדיקטים בעזרת assert על ידי הכנסת זוג הקלפים וקראיה להכנסת זוג הקלפים הבא. בסיום (Value==14) הפרדיקט מפעיל את create\_card(1,green, Serial\_num) באופן הזה אנו מכניסים לזיכרון את כל 104 הקלפים. בסיום create\_card מוסיף את שני הג'וקרים ומכניס את הפרדיקט card\_manager(106) אשר מכאן והלאה ישמור את מספר הקלפים שנותרו בקופה.

קבלת קלף מהקופה מתבצעת על ידי get\_card(card(Value,Color,Serial\_num)) המופעל על ידי get\_card(card(Value,Color)) (סידור הפרדיקטים מבטיח כניסה תחילה ל get\_card(card(Value,Color))).המחזיר את הקלף ואינו מקבל קלט. תחילה מוגרל מספר בין 1 למספר הקלפים שנותרו, המתקבל ב N על ידי קריאה ליחס card\_manager(N) הקלף שהמספר הסידורי שלו הוא הקלף שהוגרל הוא הקלף שישלח. בסיום עלינו לטפל ברשימת הקלפים לצורך המשך המשחק, לשם כך נחלק למקרים:

1. הקלף שנלקח הוא הקלף ה N -י במקרה זה נסיר את הקלף card(\_,\_,N) ונמשיך.
2. הקלף שנלקח אינו הקלף ה N-י במקרה זה אנו מסירים את הקלף שנלקח ואת הקלף ה N-י ומכניסים שוב את הקלף ה N-י לרשימת הקלפים אך הפעם עם המספר הסידורי של הקלף שהוסר.

בסיום אנו מפחיתים אחד ממספר הקלפים שבקופה על ידי הסרת היחס card\_manager(N) והכנסת היחס card\_manager(N-1).

אם מספר הקלפים שנותרו בקופה לאחר השינוי הוא 0 אנו יוצרים מחדש את רשימת הקלפים (במקרה זה אנו מוותרים על עדכון של ה card\_manager.

get\_ten\_card(List1,List2,N) – קורה 2xN פעמים ל get\_card ויוצר שני רשימות קלפים, כל רשימה מכילה N קלפים שהתקבלו מ get\_card.

**פרדיקטים נוספים:**

quicksort1(List,Sorted-Jokers-Num)

מקבל רשימת קלפים ומחזיר את הקלפים ממוינים בסדר עולה, את רשימת הג'וקרים, ואת מספר הקלפים ברשימת List. אנלוגי למיון מהיר עם רשימת הפרש המובא במדריך הלמידה עמוד 57 ההבדל הוא בטיפול בג'וקר. איבר הציר לא יכול להיות ג'וקר, כאשר האיבר הראשון הוא ברשימה הוא ג'וקר הוא מוכנס לרשימת הג'וקרים. ב partition1 כל הקלפים קטנים מג'וקר.

merge1(List1,List2,List3) - מקבל שני רשימות ממוינות ומאחד אותם.

conc2(Temp, Lonely,Result) -מעתיק את הקלף הראשון של כל זוג ב Temp ל Lonely, מחזיר את Result המכילה את Lonely ואת הקלף הראשון של כל זוג ב- Temp.