שם: יהונתן ברנסון

ת"ז: 322717232

בית ספר: תיכון רוטברג, רמת השרון

מנחה: ענבר מור

שנת 2018

מבוא

פרויקט זה עוסק בבניית אלגוריתם המשחק את המשחק רמי לתפארת.

יום שישי אחד בלילה אחרי הארוחות קבענו כמה חברים לשחק משחקי קופסה. אני ממש אוהב את המשחק רמיקוב אז שיחקנו את זה. רמיקוב הוא משחק אסטרתגי הדורש הרבה מאוד יכולת לדמיין את הלוח כמה צעדים קדימה. הוא דורש ראיה מורכבת מאוד של פירוק והרכבת סדרות. במהלך המשחק היה לי תור מורכב מאוד, ראיתי שיש לי מהלך גאוני אבל היה לי קשה להסתכל כל כך הרבה צעדים קדימה. פתאום! חשבתי לעצמי איזה מדהים אם אוכל לתכנת אלגוריתם שיעשה את זה בשבילי. הרי לי ההסתכלות המירבית קדימה היא קשה מאוד אבל מחשב עושה את בלי בעיה ובפחות משניה. החלטי לקחת את האתגר ולתכנת אלגוריתם שעושה בדיוק זה.

מדריך לשחקן

הפעלת המשחק

אופן הפעלת המשחק הינו פשוט מאוד. לוחצים על קובץ ה.exe- ומתחילים לשחק.

הוראות המשחק

**מטרת המשחק:** להיות הראשון שהוריד את כל הקלפים שלו מהתושבת, שילב אותם ברצפים או בקבוצות. תמרונים מבריקים, חשיבה אסטרטגית בהירה, מהלכים חכמים, מיומנות ומעט מזל – יביאו אתכם לניצחון!

יוצרים קבוצות של 3 או יותר קלפים של:

1. אותו מספר סוג שונה
2. סדרה עולה מאותו סוג קלף

**מספר משתתים:**1 מול המחשבאו 2 אחד מול השני.

**מי המנצח**

המנצח הוא הראשון שהוריד את כל הקלפים מהתושבת שלו ושילב אותם ברצפים או קבוצות.

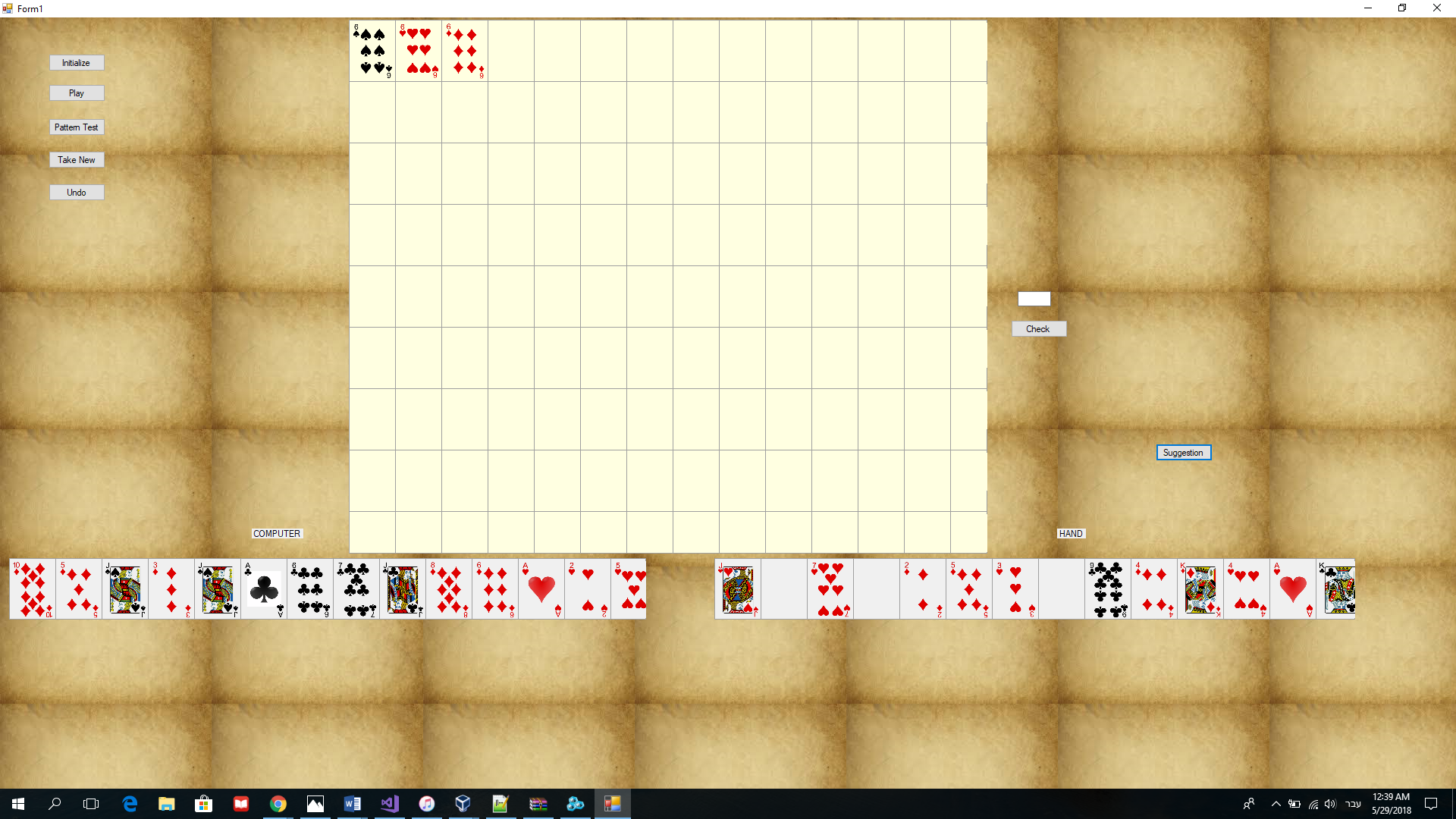
**מה השמחק כולל?**

מכיל 104 קלפים, 8 סדרות ממוספרות מ-1 עד 13 בארבעה צבעים שונים.

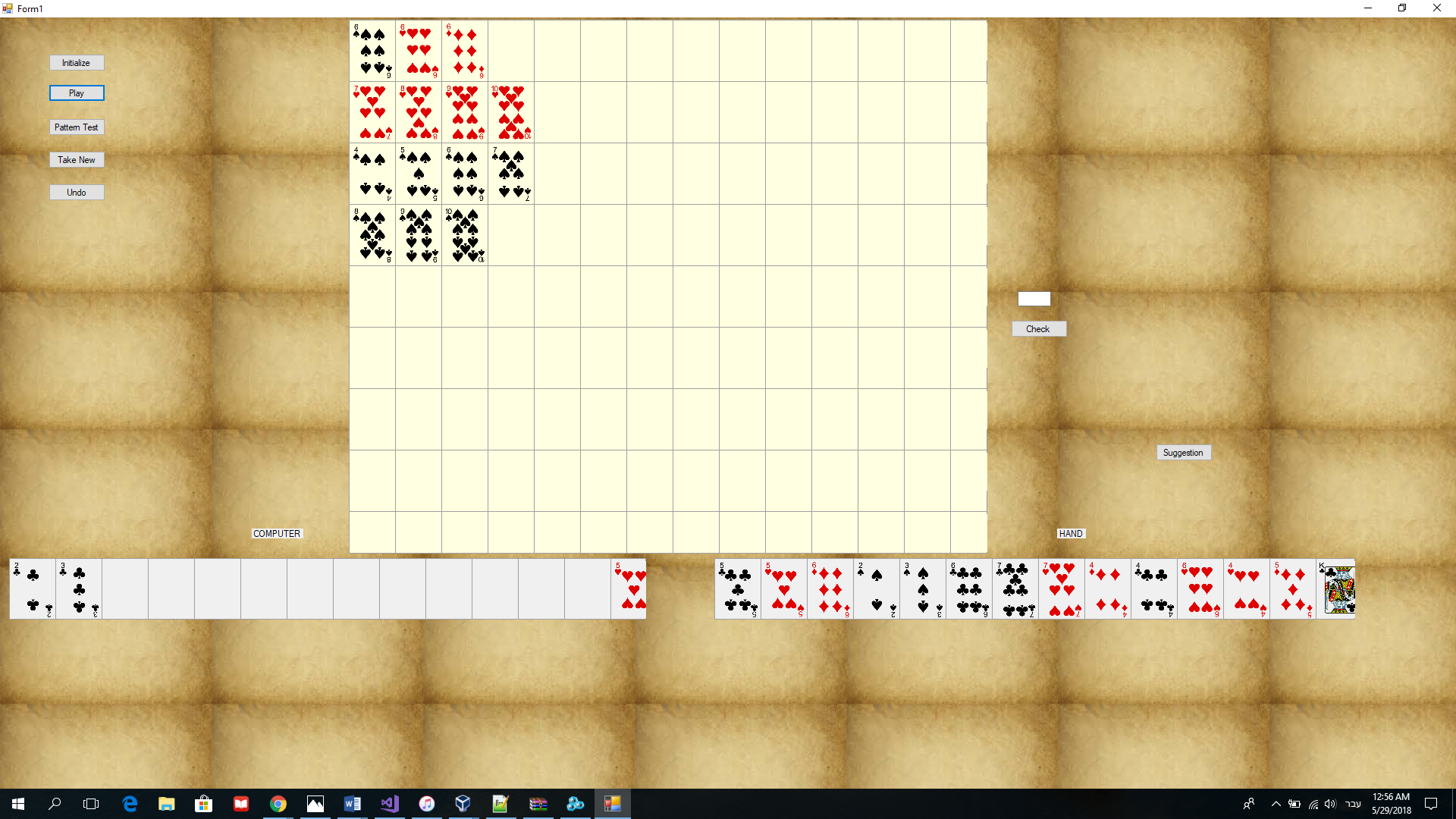
איך משחקים?

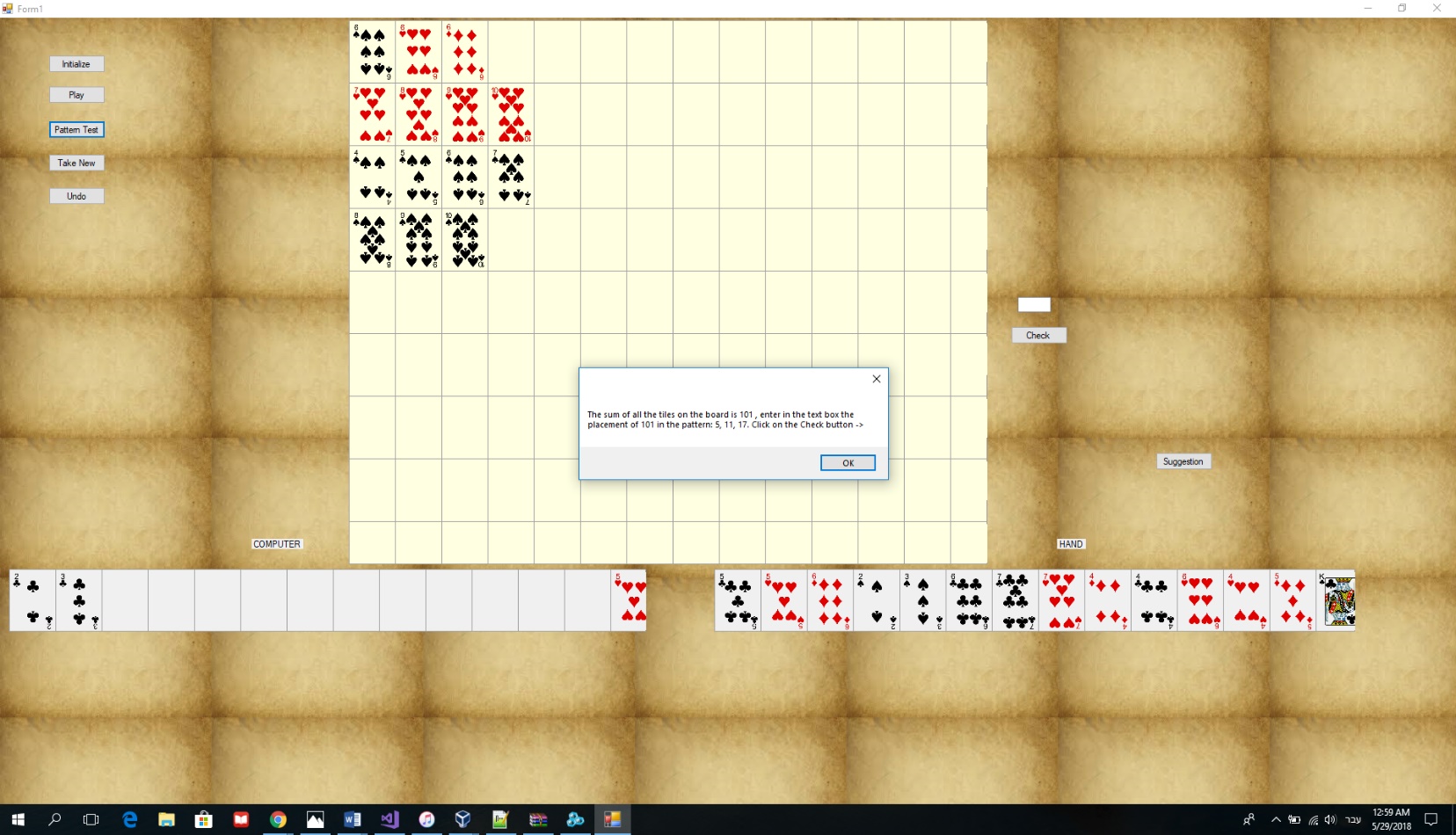
משחקים על פי כללי המשחק. השחקן משחק את התור הראשון. הוא גורר את הקלפים מהיד שלו ללוח ובכך מרכיב סדרות. ---->

כאשר השחקן סיים לשחק את תורו הוא ילחץ על כפתור ה Play-(המסומן בעיגול בתמונה) ובככך המחשב ישחק את תורו. כל אחד ישחק בתורו עד שהקלפים של מישהו יגמרו.

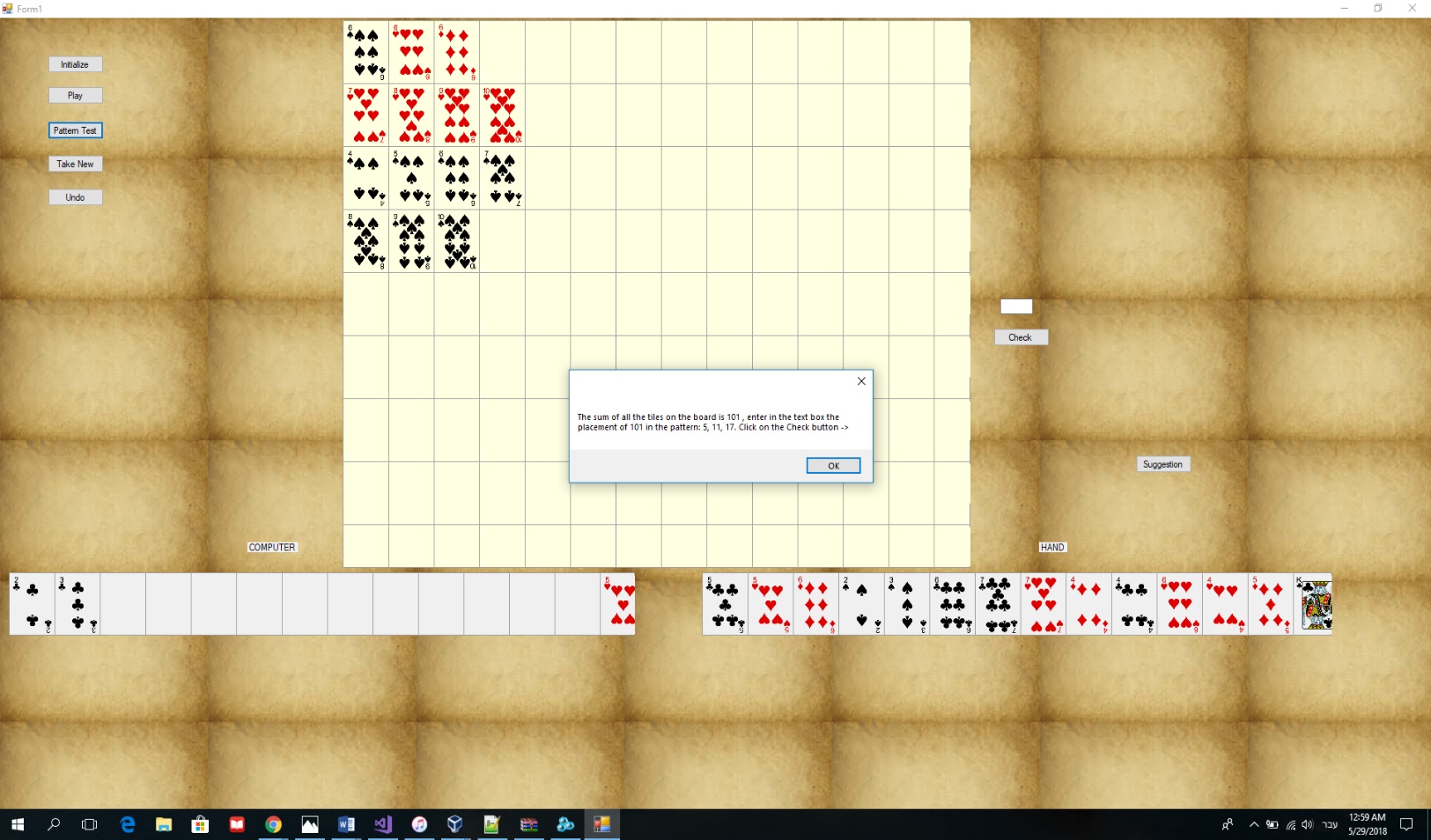


במידה ולשחקן לא יהיו כלפים לחשק באותו התור הוא ילחץ על כפתור ה Take New- (המסומן בעיגול בתמונה) ובכך יקבל קלף חדש אל ידו.



במידה והשחקן מתחרט על התור שלו, יכול ללחוץ על הפתור הUndo- (המסומן בעיגול), ולחזור תור אחד אחורה.

במידה והשחקן רוצה לבחון את החשיבה שלו באמצעות בעיות סדרה חשבוניות, יבחר בכפתור הPattern Test-. לאחר שלחץ בכפתור זה ישאל שאלה בסדרות חשבוניות הכשור לסכום ערכי הקלפים על הלוח, השחקן יקליד את תשובתו בתיבת טקסט, ילחץ על כפתור ה check-, ויבדוק אם צדק!



מחלקות

Algorithm – מכיל את האלגוריתם המשחק את המשח רמיקוב. המחלקה מכילה פעולת עזר לאלגוריתם העיקרי ביניהם יצירת סדרת צבע, יצירה סדרה עולה, והוספת קלף לסדרה בתחילתה או בסופה.

Bros – מחלקה השמשת עבורי כמבנה נתונים שיצרתי המסייעת לאלגוריתם.

Form1 – מחלקה הכוללת את כל המימוש של ממשק המשתש.

Game – מחלקה הכוללת בתוכה את ה"קערה" המכילה בתוכה את הקלפים במשחק, את יד השחקן ויד המחשב. המחלקה מסייעת בהתחול המשחק ומונעת את החזרה המירבית של אותו הקלף.

MathProject – מחלקה המסייעת למבחן הסדרות החשבוניות, היא כוללת בתוכה מספר רשימות של סדרות.

Tile –מחלקה המייצגת קלף, היא מכילה את הערך המספרי של הקלף, את סוג הקלף ותמונה המציגה את הקלף.

Type – מחלקת enum פשוטה הכוללת את ארבעה סוגי הקלפים: hearts,spades,diamonds,clubs.

Game.txt- קובץ טקסט. תכונה נוספת של האפליקציה היא שעל מנת לראות את יעילות האלגוריתם יצרתי קובץ txt הנוצר מחדש בכל הפעלת משחק ונכתב בו איך נראה הלוח לפני ואחרי שהאלגוריתם רץ. האפליקתיה כותבת ישירות לקובץ.

**מסכים:**

יש מסך אחד במשחק שהוא המשחק עצמו.

Algorithm

/// <summary>

/// Main Algorithm of the project it plays a move and tries to get rid of as much tiles it can from the playing hand

/// </summary>

/// <param name="board"></param>

/// <param name="hand"></param>

/// <param name="tile"></param>

public static void amen(Tile tile, List<Tile> comp, List<List<Tile>> board)

/// <summary>

/// Creates a color group given a tile and the tiles on the board

/// </summary>

/// <param name="tile"></param>

/// <param name=" inputhand "></param>

/// <param name=" inputboard "></param>

/// <returns> Bros type </returns>

public static Bros createColorGroup(Tile tile, List<Tile> inputhand, List<List<Tile>> inputboard)

/// <summary>

/// Creates a straight group given a tile and the tiles on the board

/// </summary>

/// <param name="tile"></param>

/// <param name=" inputhand "></param>

/// <param name=" inputboard "></param>

/// <returns> Bros type </returns>

public static Bros createStraightGroup(Tile tile, List<Tile> inputhand, List<List<Tile>> inputboard)

/// <summary>

/// checks to see if it can add a tile to the beginning or end of any group on /// the board

/// </summary>

/// <param name="board"></param>

/// <param name="hand"></param>

public static void Easy(List<List<Tile>> board, List<Tile> hand)

Bros

/// <summary>

/// data type for algorithm purposes consisting of board, hand, and group-either color or straight

/// </summary>

public class Bros

public List<List<Tile>> board = new List<List<Tile>>();//tiles on board

public List<Tile> hand = new List<Tile>(); //tiles in hand-either player or computer

public List<Tile> group = new List<Tile>(); //group either straight or color

/// <summary>

/// constructor

/// </summary>

/// <param name="board"></param>

/// <param name="hand"></param>

/// <param name="group"></param>

public Bros (List<List<Tile>> board, List<Tile> hand, List<Tile> group)

Game

/// <summary>

/// class game consists of all the tile cards in the bowl, in the player's hand and in computer's hand

/// </summary>

public class Game

public List<Tile> bowl = new List<Tile>(104); //bowl consisting of 2 deck of cards, 53 cards in each deck with joker

public List<Tile> hand = new List<Tile>(14); //cards in hand

public List<Tile> computer = new List<Tile>(14); //cards in bowl

/// <summary>

/// constructs the bowl with all cards, then gives 14 cards at random to player and computer each

/// </summary>

public Game ()

Type

//types of cards

public enum Type

{

CLUBS,DIAMONDS,SPADES,HEARTS

}

MathProject

/// <summary>

/// the class math project contains several pattern lists for the pattern tests

/// </summary>

public class MathProject

/// <returns>string containing the first 3 elements of pattern list</returns>

public static String ListToString (List<int> lst)

Game.txt example:

---------------------COMPUTER TURN---------------------

------------------------BEFORE------------------------

------------------------AFTER------------------------

(HEARTS 7)(HEARTS 8)(HEARTS 9)(HEARTS 10)

(SPADES 4)(SPADES 5)(SPADES 6)(SPADES 7)(SPADES 8)(SPADES 9)(SPADES 10)

---------------------PLAYER TURN---------------------

------------------------BEFORE------------------------

(HEARTS 7)(HEARTS 8)(HEARTS 9)(HEARTS 10)

(SPADES 4)(SPADES 5)(SPADES 6)(SPADES 7)

(SPADES 8)(SPADES 9)(SPADES 10)

------------------------AFTER------------------------

(HEARTS 7)(HEARTS 8)(HEARTS 9)(HEARTS 10)

(SPADES 5)(SPADES 6)(SPADES 8)(SPADES 9)(SPADES 10)

(CLUBS 5)(HEARTS 5)(DIAMONDS 5)

(SPADES 2)(SPADES 3)(CLUBS 7)(HEARTS 7)(SPADES 7)

(CLUBS 4)(DIAMONDS 4)(HEARTS 4)(SPADES 4)

(DIAMONDS 6)(CLUBS 6)(HEARTS 6)

Form1

/// <summary>

/// initializes the board. Gives computer hand and player hand each 14 cards.

/// </summary>

public Form1()

/// <summary>

/// Adds an event handler to all tiles on the board, making it possible to move them around with mouse clicks.

/// </summary>

private void AddMouseEventHandlerToBoard()

/// <summary>

/// Adds an event handler to all tiles in player hand, making it possible to move them around with mouse clicks.

/// </summary>

private void AddMouseEventHandlerToHand()

/// <summary>

/// Adds an event handler to all tiles in computer hand, making it possible to move them around with mouse clicks.

/// </summary>

private void AddMouseEventHandlerToComputer()

/// <summary>

/// Adds a blank picture box to all boxes in table making it possible to use control methods

/// </summary>

private void fillBoard ()

/// <summary>

/// button initilializes player hand and computer hand with good tile cards for a more enjoyable game

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void initialize\_Click(object sender, EventArgs e)

/// <summary>

/// button that plays turn for computer

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void play\_Click(object sender, EventArgs e)

/// <summary>

/// special feature. My math teacher asked us to build a

/// math game for a better learning experience for students.

/// I added this button which askes you a math question in patterns

/// based on the sum of the tiles on the board.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pattern\_Click(object sender, EventArgs e)

/// <summary>

/// checks if the answer to the pattern test

/// question is correct and pops a message box with

/// either "correct" or "wrong"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void checkBtn\_Click(object sender, EventArgs e)

/// <summary>

/// gets the tile that was clicked on and inserts it into CardPictureBox

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void handclickOnSpace(object sender, MouseEventArgs e)

/// <summary>

/// pastes the tile onto the board

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void boardclickOnSpace(object sender, MouseEventArgs e)

/// <summary>

/// Method that updates the backend of the board by the UI

/// </summary>

public int UpdateBoard()

/// <summary>

/// gives player hand random new card from bowl

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void takeNewBtn\_Click(object sender, EventArgs e)

/// <summary>

/// allows you to undo a move

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void undoBtn\_Click(object sender, EventArgs e)

/// <summary>

/// suggestion button for the player. plays the move for him.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void suggestBtn\_Click(object sender, EventArgs e)

/// <summary>

/// checks if either the player or the computer wins the game.

/// if so, pops a message box with whomever won.

/// </summary>

private void WinOrLose()

/// <summary>

/// method that adds tiles to computer table. It connects between the backend to the UI.

/// </summary>

/// <param name="comp"></param>

public void compTB(List<Tile> comp)

/// <summary>

/// method that adds tiles to player table. It connects between the backend to the UI.

/// </summary>

/// <param name="hand"></param>

public void handTB(List<Tile> hand)

/// <summary>

/// method that adds the board list to the board table. It connects between the backend to the UI.

/// </summary>

/// <param name="board"></param>

/// <param name="row"></param>

public void boardTB(List<List<Tile>> board, int row)

/// <summary>

/// helper method to boardTB, adds individual groups to board

/// </summary>

/// <param name="group"></param>

/// <param name="row"></param>

/// <param name="column"></param>

public void groupTB(List<Tile> group, int row, int column)

בעיות ופתרונן

האמת היא שאם אני רוצה אני מסוגל לכתוב ספר שלם רק על הבעיות של הפרוייקט...לא היה חסר בכלל!!

אבל אם צריך לפרט כמה ספציפיות אז אגוון וא

ת מכמה תחומים:

הבעיה הראשונה

סיכום

תכנות הפרויקט הייתה עבורי חוויה מהנה ומלמדת מאוד. חוץ מהידע שרכשתי השנה בשפת התכנות #C וסביבת הפיתוח winformsשל visual studios למדתי השנה בעיקר איך ללמוד לבד. גיליתי שהמורה הכי טוב הוא גוגל ושהדרך הכי טובה ללמוד לתכנת ולהתבגר על המכשולים המוצבים היא ללמוד ממפתחים אחרים כמוני שנתקלו באותם בעיות ומאנשים אחרים שעזרו להם להתגבר על בעיות אלו.

בנוסף למשחבים הפרויקט גרם לי לחשוב על בעיות מורכבות באלגוריתמיקה ולהפעיל מאוד את החשיבה שלי.

אני מרוצה מאוד מבחירתי בפרויקט זה וחושב כי הפרויקט שיפר את יכולותיי בתחום התכנות בכלל!

תודות

יצא לי השנה לעבוד עם ענבר מור שהוא מנחה מדהים עם ידע רחב שעזר לי בכל בעיה שנתקלתי בה. תודה גם לעינת לוי, רכזת המגמה, על התמיכה הגדול ועל כך שלימדה אותי שנתיים לפני הפרויקט ובזכותה אני יודע לתכנת.

ארצה להודות לבית הספר, תיכון רוטברג למדעים ולאומנויות, אשר השקיע במעבדות מחשבים מתקדמות על מנת שתהיה לנו נוחות עבודה אופטימלית.

