|  |  |
| --- | --- |
| CANDY CRUSH-PF PROJECT  **SUBMITTED TO: SIR MOIZ QADIR** | 🡨--**INTRO**--🡪  **A simplified version of candy crush games copy that u can play on console(All Studied concepts used).**  **Muhammad ABDULLAH**  **23F-0581**  **MUHAMMAD QASIM**  **23F-0685** |

**<----- CANDY CRUSH CODE---->**

**//// COMMENTS EXPLAINING FUNCTIONALITY!**

**//**

**// /\***

**/\*This C++ program implements a simplified version of the Candy Crush game.**

**The game revolves around a grid of candies, and the objective is to swap adjacent**

**candies to create matches of three or more candies in a row or column.When a match is**

**formed, those candies are removed, and new candies fall from the top to fill the gaps**

**.The game continues until no more matches are possible.The implementation employs a 2D array**

**to represent the candy grid, and functions are defined to handle swapping, checking for matches,**

**and updating the grid state.The game loop manages user input, grid updates, and score tracking.This well**

**- organized and modular design enables easy expansion or modification of game features, making it an ideal**

**foundation for building upon or customizing the Candy Crush game.\*/**

**// -------------------------------CODE------------------------------------**

**#include <iostream>**

**#include <string>**

**#include <cstdlib>**

**#include <ctime>**

**#include <iomanip>**

**#include <Windows.h>**

**#include <fstream>**

**#include<stdlib.h>**

**#include<conio.h>**

**#include <chrono>**

**#include <thread>**

**using namespace std;**

**// -------------------------- DEFINING KEYS-----------------------------**

**#define KEY\_ENTER 13**

**#define KEY\_UP 72**

**#define KEY\_DOWN 80**

**#define KEY\_LEFT 75**

**#define KEY\_RIGHT 77**

**// -------------------------SETTING COLOURS-----------------------------**

**const string ch1 = "\033[31m#\033[0m";**

**const string ch2 = "\033[34m@\033[0m";**

**const string ch3 = "\033[32m&\033[0m";**

**const string ch4 = "\033[35m%\033[0m";**

**const string ch5 = "\033[33m\*\033[0m";**

**const string ch6 = "\033[36m+\033[0m";**

**const string ch7 = "\033[37m?\033[0m";**

**// -------------------------Global variables-----------------------------**

**int moves = 15;**

**int score = 0;**

**string name;**

**const int easyTimeLimit = 60; // 60 seconds for easy mode**

**const int hardTimeLimit = 45; // 45 seconds for hard mode**

**//string array for 5 characters easy mode**

**const string ch\_array[] = { ch1, ch2, ch3, ch4, ch5 };**

**//string array for 7 character hard mode**

**const string ch\_array\_h[] = { ch1, ch2, ch3, ch4, ch5, ch6, ch7 };**

**//consecutiveLimit is a variable to use in isvalid funstion as a limit**

**const int consecutiveLimit = 2;**

**//-------------------------PROTOTYPES---------------------------**

**void easy\_board();**

**void hard\_board();**

**void aftergame(int score);**

**void printBoard();**

**string crushrand(string candy);**

**string crushrandhard(string candy);**

**void swpspaceHARD(string arr[][10]);**

**void fill(string arr[8][8]);**

**void aftergame(int score) {**

**if (score > 450)**

**{**

**cout << "YOU WIN!" << endl;**

**cout << " \* \*\* \*\n";**

**cout << " \* \*\* \*\n";**

**cout << " \* \*\* \*\n";**

**cout << " \* \*\* \*\n";**

**cout << " \* \*\* \*\n";**

**cout << " \* \*\* \*\n";**

**cout << " \* \*\* \*\n";**

**cout << " \* \* \* \*\n";**

**cout << " \* \* \* \*\n";**

**cout << " \* \* \* \*\n";**

**cout << " \* \* \* \* \n";**

**cout << " \* \* \n \n\n";**

**}**

**else**

**{**

**cout << "YOU LOSE" << endl;**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\*\n";**

**cout << " \*\*\* \* \* \* \* \* \* \* \*\n";**

**cout << " \*\* \* \* \* \* \* \* \* \* \*\n\n\n";**

**}**

**}**

**string crushrand(string candy) { // creates random key in 2D-array easy mode**

**int c;**

**c = rand() % 5 + 1;**

**if (c == 1)**

**candy = ch1;**

**else if (c == 2)**

**candy = ch2;**

**else if (c == 3)**

**candy = ch3;**

**else if (c == 4)**

**candy = ch4;**

**else if (c == 5)**

**candy = ch5;**

**return candy;**

**}**

**string crushrandhard(string candy) { // creates random key in 2D-array easy mode**

**int c;**

**c = (rand() % 7) + 1;**

**if (c == 1)**

**candy = ch1;**

**else if (c == 2)**

**candy = ch2;**

**else if (c == 3)**

**candy = ch3;**

**else if (c == 4)**

**candy = ch4;**

**else if (c == 5)**

**candy = ch5;**

**else if (c == 6)**

**candy = ch6;**

**else if (c == 7)**

**candy = ch7;**

**return candy;**

**}**

**void fill(string arr[8][8]) // check if array element has space value and replace with random candyy it is executed after swpspaceeasy function**

**{**

**string candy = " ";**

**for (int x = 0; x < 8; x++)**

**{**

**for (int y = 0; y < 8; y++)**

**{**

**if (arr[x][y] == " ")**

**{**

**arr[x][y] = crushrand(candy);**

**if (arr[x][y] == arr[x][y + 1] && arr[x][y] == arr[x][y + 2] ||//column wise horizontal**

**arr[x][y] == arr[x + 1][y] && arr[x][y] == arr[x + 2][y] || // row wise vertical**

**arr[x][y] == arr[x - 1][y + 1] && arr[x][y] == arr[x + 1][y - 1] || //forward diagonal**

**arr[x][y] == arr[x - 1][y - 1] && arr[x][y] == arr[x + 1][y + 1]) //backward diagonal**

**{**

**char c = ' ';**

**arr[x][y] = crushrand(candy);**

**}**

**}**

**}**

**}**

**}**

**void swpspaceHARD(string arr[][10])//swap spaces to top in hard mode**

**{**

**for (int i = 0; i < 10; i++)**

**{**

**for (int j = 0; j < 10; j++)**

**{**

**if (arr[i][j] == " ")**

**{**

**int l = i, m = j;**

**while (l > 0)**

**{**

**arr[l][m] = arr[l - 1][m];**

**l--;**

**}**

**}**

**}**

**}**

**}**

**void fillhard(string arr[10][10]) // check if array element has space value and replace with random candy it is executed after swpspacehard function**

**{**

**string candy = " ";**

**for (int x = 0; x < 10; x++)**

**{**

**for (int y = 0; y < 10; y++)**

**{**

**if (arr[x][y] == " ")**

**{**

**arr[x][y] = crushrandhard(candy);**

**if (arr[x][y] == arr[x][y + 1] && arr[x][y] == arr[x][y + 2] ||//column wise horizontal**

**arr[x][y] == arr[x + 1][y] && arr[x][y] == arr[x + 2][y] || // row wise vertical**

**arr[x][y] == arr[x - 1][y + 1] && arr[x][y] == arr[x + 1][y - 1] || //forward diagonal**

**arr[x][y] == arr[x - 1][y - 1] && arr[x][y] == arr[x + 1][y + 1]) //backward diagonal**

**{**

**arr[x][y] = crushrandhard(candy);**

**}**

**}**

**}**

**}**

**}**

**void swpspaceEASY(string arr[8][8]) //swap spaces to top in easy mode**

**{**

**for (int i = 0; i < 8; i++)**

**{**

**for (int j = 0; j < 8; j++)**

**{**

**if (arr[i][j] == " ")**

**{**

**int l = i, m = j;**

**while (l > 0)**

**{**

**arr[l][m] = arr[l - 1][m];**

**l--;**

**}**

**}**

**}**

**}**

**}**

**bool isValidNeighbor(int selectedRow, int selectedCol, int currentRow, int currentCol) {**

**// Check if the candies are neighbors (up, down, left, or right)**

**int rowDiff = abs(selectedRow - currentRow);**

**int colDiff = abs(selectedCol - currentCol);**

**return (rowDiff == 1 && colDiff == 0) || (rowDiff == 0 && colDiff == 1);**

**}**

**void check\_easy(string arr[][8], int i, int j) {**

**int move = 15;**

**while (move > 0)**

**{**

**for (int i = 0; i < 8; i++)**

**{**

**for (int j = 0; j < 8; j++)**

**{**

**if (arr[i][j] == arr[i + 1][j] && arr[i][j] == arr[i + 2][j] && arr[i][j] == arr[i + 2][j + 1] && arr[i][j] == arr[i + 2][j + 2])//L shape Condition spaces**

**{**

**arr[i][j] = ' ';**

**arr[i + 1][j] = ' ';**

**arr[i + 2][j] = ' ';**

**arr[i][j + 1] = ' ';**

**arr[i][j + 2] = ' ';**

**score += 3;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i + 1][j] && arr[i][j] == arr[i + 2][j])// vertical condition check**

**{**

**arr[i][j] = ' ';**

**arr[i + 1][j] = ' ';**

**arr[i + 2][j] = ' ';**

**score += 1;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i][j + 1] && arr[i][j] == arr[i][j + 2]) // horizontal condition**

**{**

**arr[i][j] = ' ';**

**arr[i][j + 1] = ' ';**

**arr[i][j + 2] = ' ';**

**score += 1;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i - 1][j + 1] && arr[i][j] == arr[i + 1][j - 1]) // forward diaginal condition**

**{**

**arr[i][j] = ' ';**

**arr[i - 1][j + 1] = ' ';**

**arr[i + 1][j - 1] = ' ';**

**score += 2;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i - 1][j - 1] && arr[i][j] == arr[i + 1][j + 1])// backward diagonal condition**

**{**

**arr[i][j] = ' ';**

**arr[i - 1][j - 1] = ' ';**

**arr[i + 1][j + 1] = ' ';**

**score += 2;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**}**

**}**

**move--;**

**swpspaceEASY(arr);**

**fill(arr);**

**}**

**}**

**void check\_hard(string arr[][10], int i, int j) {**

**int move = 15;**

**while (move != 0)**

**{**

**for (int i = 0; i < 10; i++)**

**{**

**for (int j = 0; j < 10; j++)**

**{**

**int c = 0;**

**if (arr[i][j] == arr[i + 1][j] && arr[i][j] == arr[i + 2][j] && arr[i][j] == arr[i + 2][j + 1] && arr[i][j] == arr[i + 2][j + 2])//L shape Condition spaces**

**{**

**arr[i][j] = ' ';**

**arr[i + 1][j] = ' ';**

**arr[i + 2][j] = ' ';**

**arr[i][j + 1] = ' ';**

**arr[i][j + 2] = ' ';**

**score += 3;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i + 1][j] && arr[i][j] == arr[i + 2][j])// vertical condition check**

**{**

**arr[i][j] = ' ';**

**arr[i + 1][j] = ' ';**

**arr[i + 2][j] = ' ';**

**score += 1;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i][j + 1] && arr[i][j] == arr[i][j + 2]) // horizontal condition**

**{**

**arr[i][j] = ' ';**

**arr[i][j + 1] = ' ';**

**arr[i][j + 2] = ' ';**

**score += 1;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i - 1][j + 1] && arr[i][j] == arr[i + 1][j - 1]) // forward diaginal condition**

**{**

**arr[i][j] = ' ';**

**arr[i - 1][j + 1] = ' ';**

**arr[i + 1][j - 1] = ' ';**

**score += 20;**

**score += 1;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**else if (arr[i][j] == arr[i - 1][j - 1] && arr[i][j] == arr[i + 1][j + 1])// backward diagonal condition**

**{**

**arr[i][j] = ' ';**

**arr[i - 1][j - 1] = ' ';**

**arr[i + 1][j + 1] = ' ';**

**score += 1;**

**system("CLS");**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tSweet\033[0m" << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\tLOADING\033[0m" << endl;**

**Sleep(5);**

**system("CLS");**

**}**

**}**

**}**

**move--;**

**swpspaceHARD(arr);**

**fillhard(arr);**

**}**

**}**

**void displayArray(string array[][8], int selectedRow, int selectedCol) {**

**system("cls"); // Clear the console screen**

**cout << endl;**

**//to display a message of player and mode choosed by him/her.**

**cout << "\033[31m\t\t\t\t\t\t " << name << " Choose EASY MODE\033[0m\n";**

**cout << "\033[31m\t\t\t\t\t\t ====================\033[0m\n";**

**cout << "\n\n\n";**

**for (int i = 0; i < 8; ++i) {**

**cout << "\t\t\t\t";**

**for (int j = 0; j < 8; ++j) {**

**if (i == selectedRow && j == selectedCol) {**

**cout << "(x) | ";**

**}**

**else {**

**cout << " " << array[i][j] << " | ";**

**}**

**}**

**cout << endl;**

**cout << "\t\t\t\t------------------------------------------------";**

**cout << endl;**

**cout << '\n';**

**}**

**cout << "\n\n\t\tScore: " << score << "\t\tMoves: " << moves;**

**cout << "\n\n\033[31m Press Zero for shuffling the board if there is no match left!\033[0m" << endl;**

**}**

**void swape(string board[][8]) {**

**int currentRow = 0;**

**int currentCol = 0;**

**char key;**

**int selectedRow = 0;**

**int selectedCol = 0;**

**bool candySelected = false;**

**string temp1;**

**std::chrono::seconds duration(60); // TIME LIMIT**

**auto startTime = std::chrono::high\_resolution\_clock::now();**

**do {**

**for (int i = 0; i < 5; ++i) {**

**displayArray(board, currentRow, currentCol);**

**key = \_getch(); // Get the key without waiting for Enter**

**switch (key) {**

**case KEY\_UP: // Up arrow key**

**if (currentRow > 0) {**

**--currentRow;**

**}**

**break;**

**case KEY\_DOWN: // Down arrow key**

**if (currentRow < 8 - 1) {**

**++currentRow;**

**}**

**break;**

**case KEY\_LEFT: // Left arrow key**

**if (currentCol > 0) {**

**--currentCol;**

**}**

**break;**

**case KEY\_RIGHT: // Right arrow key**

**if (currentCol < 8 - 1) {**

**++currentCol;**

**}**

**break;**

**case KEY\_ENTER: // Select or swap candy**

**if (!candySelected) {**

**// First press of ENTER - select candy**

**selectedRow = currentRow;**

**selectedCol = currentCol;**

**candySelected = true;**

**}**

**else {**

**}**

**break;**

**case 's': // Swap candy when 's' key is pressed**

**if (candySelected && isValidNeighbor(selectedRow, selectedCol, currentRow, currentCol)) {**

**// Swap only if the selected candy and the current candy are neighbors**

**swap(board[selectedRow][selectedCol], board[currentRow][currentCol]);**

**check\_easy(board, currentRow, currentCol);**

**/\* if (moves==0)**

**{**

**aftergame(score);**

**}\*/**

**moves--;**

**}**

**candySelected = false;**

**break;**

**case '0':**

**easy\_board();**

**break;**

**}**

**auto currentTime = std::chrono::high\_resolution\_clock::now();**

**auto elapsedTime = std::chrono::duration\_cast<std::chrono::seconds>(currentTime - startTime);**

**// Check if the time limit is reached**

**if (elapsedTime >= duration) {**

**system("cls");**

**cout << "\033[31m\t\t\t\t\t\t\t\tTimes Up!!\033[0m" << endl;**

**cout << "\033[31m\t\t\t\t\t\t\t\tYour score: \033[0m" << score << endl;**

**cout << "\033[31m\t\t\t\t\t\t\t\tPlayer Name\033[0m" << name << endl;**

**Sleep(5000);**

**system("cls");**

**cout << "Program is exiting." << endl;**

**exit(5);**

**}**

**}**

**} while (moves != 0); // Press 'x' to exit**

**}**

**void displayArray1(string array[][10], int selectedRow, int selectedCol) {**

**system("cls");**

**cout << endl;// Clear the console screen**

**//to display a message of player and mode choosed by him/her.**

**cout << endl;**

**cout << "\033[31m\t\t\t\t\t\t " << name << " Choose HARD MODE\033[0m\n";**

**cout << "\033[31m\t\t\t\t\t\t ====================\033[0m\n";**

**cout << "\n\n\n";**

**for (int i = 0; i < 10; ++i) {**

**cout << "\t\t\t\t";**

**for (int j = 0; j < 10; ++j) {**

**if (i == selectedRow && j == selectedCol) {**

**cout << "(+) | ";**

**}**

**else {**

**cout << " " << array[i][j] << " | ";**

**}**

**}**

**cout << endl;**

**cout << "\t\t\t\t------------------------------------------------------------";**

**cout << endl;**

**cout << '\n';**

**}**

**cout << "\n\n\t\tScore: " << score << "\t\tMoves: " << moves;**

**cout << "\n\n\033[31m Press Zero for shuffling the board if there is no match left!\033[0m" << endl;**

**}**

**void swaph(string board[][10]) {**

**int currentRow = 0;**

**int currentCol = 0;**

**char key;**

**int selectedRow = 0;**

**int selectedCol = 0;**

**bool candySelected = false;**

**string temp1;**

**std::chrono::seconds duration(40); // TIME LIMIT**

**auto startTime = std::chrono::high\_resolution\_clock::now();**

**do {**

**displayArray1(board, currentRow, currentCol);**

**key = \_getch(); // Get the key without waiting for Enter**

**switch (key) {**

**case KEY\_UP: // Up arrow key**

**if (currentRow > 0) {**

**--currentRow;**

**}**

**break;**

**case KEY\_DOWN: // Down arrow key**

**if (currentRow < 10 - 1) {**

**++currentRow;**

**}**

**break;**

**case KEY\_LEFT: // Left arrow key**

**if (currentCol > 0) {**

**--currentCol;**

**}**

**break;**

**case KEY\_RIGHT: // Right arrow key**

**if (currentCol < 10 - 1) {**

**++currentCol;**

**}**

**break;**

**case KEY\_ENTER: // Select or swap candy**

**if (!candySelected) {**

**// First press of ENTER - select candy**

**selectedRow = currentRow;**

**selectedCol = currentCol;**

**candySelected = true;**

**}**

**else {**

**// Second press of ENTER - navigate for swapping**

**// Do nothing here, just wait for arrow key or 's' key**

**}**

**break;**

**case 's': // Swap candy when 's' key is pressed**

**if (candySelected && isValidNeighbor(selectedRow, selectedCol, currentRow, currentCol)) {**

**// Swap only if the selected candy and the current candy are neighbors**

**swap(board[selectedRow][selectedCol], board[currentRow][currentCol]);**

**check\_hard(board, currentRow, currentCol);**

**moves--;**

**}**

**candySelected = false;**

**break;**

**case '0':**

**hard\_board();**

**break;**

**}**

**//Time counter**

**auto currentTime = std::chrono::high\_resolution\_clock::now();**

**auto elapsedTime = std::chrono::duration\_cast<std::chrono::seconds>(currentTime - startTime);**

**// Check if the time limit is reached**

**if (elapsedTime >= duration) {**

**system("cls");**

**cout << "\033[31m\t\t\t\t\t\t\t\tTimes Up!!\033[0m" << endl;**

**cout << "\033[31m\t\t\t\t\t\t\t\tYour score: \033[0m" << score << endl;**

**cout << "\033[31m\t\t\t\t\t\t\t\tPlayer Name: \033[0m" << name << endl;**

**Sleep(5000);**

**system("cls");**

**cout << "Program is exiting." << endl;**

**exit(5);**

**}**

**} while (moves != 0); // Press 'x' to exit**

**}**

**// Function to check if placing a candy at a certain position is valid For Easy Display**

**bool isValid(string board[8][8], int row, int col, string candy) {**

**// Check horizontally**

**int countHorizontal = 0;**

**for (int i = col - consecutiveLimit; i <= col; ++i) {**

**if (i >= 0 && board[row][i] == candy) {**

**countHorizontal++;**

**}**

**else {**

**break;**

**}**

**}**

**// Check vertically**

**int countVertical = 0;**

**for (int i = row - consecutiveLimit; i <= row; ++i) {**

**if (i >= 0 && board[i][col] == candy) {**

**countVertical++;**

**}**

**else {**

**break;**

**}**

**}**

**// Check if candy appears on three or more consecutive blocks**

**return countHorizontal < consecutiveLimit && countVertical < consecutiveLimit;**

**}**

**//print board function for easy board**

**void printBoard(string board[8][8]) {**

**for (int i = 0; i < 8; i++) {**

**cout << "\t\t\t\t\t\t\t\t";**

**for (int j = 0; j < 8; j++) {**

**cout << setw(4) << board[i][j] << " | ";**

**}**

**cout << endl;**

**cout << "\t\t\t\t\t\t\t\t";**

**for (int k = 0; k < 8; k++) {**

**cout << "----";**

**}**

**cout << endl;**

**}**

**swape(board);**

**}**

**//print board function for hard board**

**void printBoard1(string board[][10]) {**

**for (int i = 0; i < 10; i++) {**

**cout << "\t\t\t\t\t\t\t\t"; //to bring Board in center**

**for (int j = 0; j < 10; j++) {**

**cout << setw(4) << board[i][j] << " | ";**

**}**

**cout << endl;**

**cout << "\t\t\t\t\t\t\t\t";**

**for (int k = 0; k < 10; k++) {**

**cout << "----";**

**}**

**cout << endl;**

**}**

**swaph(board);**

**}**

**//Easy Board Display**

**void easy\_board() {**

**system("cls");**

**srand(time(0));**

**string board[8][8];**

**for (int i = 0; i < 8; i++) {**

**for (int j = 0; j < 8; j++) {**

**int randomIndex;**

**do {**

**randomIndex = rand() % (sizeof(ch\_array) / sizeof(ch\_array[0]));**

**} while (!isValid(board, i, j, ch\_array[randomIndex]));**

**board[i][j] = ch\_array[randomIndex];**

**}**

**}**

**printBoard(board);**

**}**

**// Function to check if placing a candy at a certain position is valid For hard Board**

**bool isValid(string board[10][10], int row, int col, string candy) {**

**// Check horizontally**

**int countHorizontal = 0;**

**for (int i = col - consecutiveLimit; i <= col; ++i) {**

**if (i >= 0 && board[row][i] == candy) {**

**countHorizontal++;**

**}**

**else {**

**break;**

**}**

**}**

**// Check vertically**

**int countVertical = 0;**

**for (int i = row - consecutiveLimit; i <= row; ++i) {**

**if (i >= 0 && board[i][col] == candy) {**

**countVertical++;**

**}**

**else {**

**break;**

**}**

**}**

**// Check if candy appears on three or more consecutive blocks**

**return countHorizontal < consecutiveLimit && countVertical < consecutiveLimit;**

**}**

**//Hard board**

**void hard\_board() {**

**srand(time(0));**

**string board[10][10];**

**for (int i = 0; i < 10; i++) {**

**for (int j = 0; j < 10; j++) {**

**int randomIndex;**

**do {**

**randomIndex = rand() % (sizeof(ch\_array\_h) / sizeof(ch\_array\_h[0]));**

**} while (!isValid(board, i, j, ch\_array\_h[randomIndex]));**

**board[i][j] = ch\_array\_h[randomIndex];**

**}**

**}**

**printBoard1(board);**

**}**

**//Display credits of Game**

**void credits() {**

**ifstream my\_input\_file("Credits.txt");**

**if (!my\_input\_file.is\_open()) {**

**cout << "File cannot be opened." << endl;**

**return;**

**}**

**cout << "\n\n\n\n";**

**char ch;**

**while (my\_input\_file.get(ch)) {**

**cout << "\033[33m" << ch << "\033[0m";**

**}**

**cout << endl;**

**my\_input\_file.close();**

**}**

**void file\_write() {**

**{**

**cout << name << " scored :" << score << endl;**

**fstream file;/// ifstream fin..../// ofstream out ... fout ... fin.>>**

**file.open("Score.txt", ios::app); // fin.open//ios::in.**

**file << endl;**

**file << name;**

**file << " ";**

**file << score;**

**file << endl;**

**file.close();**

**}**

**}**

**int main() {**

**char ch = ' ';**

**char exi\_;**

**do {**

**int mode = 0, play\_game = 0;**

**cout << "\033[1m";**

**cout << endl;**

**//Display menu \033[35m" "\033[om for different colors onconsole**

**//WELCOME MESSAGE**

**cout << "\033[35m\t\t\t\t\t\t=======================\033[0m\n";**

**cout << "\033[33m\t\t\t\t\t\tWELCOME TO CANDY CRUSH\033[0m\n";**

**cout << "\033[35m\t\t\t\t\t\t=======================\033[0m\n";**

**//MENU BAR**

**cout << endl;**

**cout << "\t\t\t\t\t\t,-----------------------,\n";**

**cout << "\t\t\t\t\t\t| 1. Play Game |\n";**

**cout << "\t\t\t\t\t\t| 2. Credits |\n";**

**cout << "\t\t\t\t\t\t| 3. Instructions |\n";**

**cout << "\t\t\t\t\t\t| 4. Score section |\n";**

**cout << "\t\t\t\t\t\t| 5. Exit |\n";**

**cout << "\t\t\t\t\t\t| |\n";**

**cout << "\t\t\t\t\t\t| Your Choice |\n";**

**cout << "\t\t\t\t\t\t'-----------------------'\n";**

**cout << "\t\t\t\t\t\t\tYour Choice: ";**

**cin >> play\_game;**

**if (play\_game == 1) {**

**system("CLS");**

**cout << endl << endl << endl;**

**cout << "\t\t\t\t\t\t\033[36m| PLEASE ENTER YOUR NAME: |\033[0m" << endl;**

**cout << "\033[36m\t\t\t\t\t\t ----------------------\033[0m" << endl;**

**cout << "\t\t\t\t\t\t ";**

**cout << "NAME: ";**

**cin.ignore();// without this control ignores user to input his name..**

**getline(cin, name); // gets input in string with spaces..**

**system("cls");**

**cout << endl;**

**cout << "\033[1m"; //to bold output on console**

**cout << "\033[32m\t\t\t\t\t 1. Easy Mode\033[0m\n";**

**cout << "\033[32m\t\t\t\t\t 2. Hard Mode\033[0m\n";**

**cout << "\t\t\t\t\t ==============\n";**

**cout << "\t\t\t\t\tYour Choice: ";**

**cin >> mode;**

**system("cls");**

**if (mode == 1) {**

**cout << endl;**

**cout << "\033[33m\t\t\t\t\t\t\t\t\t\t EASY MODE\033[0m\n";**

**cout << "\033[33m\t\t\t\t\t\t\t\t\t\t =========\033[0m\n";**

**cout << "\t\t\t\t\t\t\tTime: \t\t\t\t\t\t\t Score: \n";**

**cout << endl;**

**cout << endl;**

**easy\_board();**

**Sleep(6000);**

**cout << "\t\t\t\t\t\t\t\t\t\t\t\t\t " << score;**

**system("cls");**

**cout << endl; cout << "\033[31m\t\t\t\t\t\t\t\tTIME IS OVER...\033[0m\n\n";**

**aftergame(score);**

**file\_write();**

**cin >> ch;**

**}**

**else if (mode == 2) {**

**cout << endl;**

**cout << "\033[31m\t\t\t\t\t\t\t\t\t\t HARD MODE\033[0m\n";**

**cout << "\033[31m\t\t\t\t\t\t\t\t\t\t =========\033[0m\n";**

**cout << "\t\t\t\t\t\t\tTime: \t\t\t\t\t\t\t Score: \n";**

**cout << endl;**

**cout << endl;**

**hard\_board();**

**Sleep(6000);**

**system("cls");**

**cout << endl; cout << "\033[31m\t\t\t\t\t\t\t\tTIME IS OVER...\033[0m\n\n";**

**aftergame(score);**

**file\_write();**

**cin >> ch;**

**}**

**cin >> ch;**

**}**

**// 2 will show credit of game**

**else if (play\_game == 2)**

**{**

**system("cls");**

**credits();**

**cin >> ch;**

**}**

**// 3 will show instruction by file reading**

**else if (play\_game == 3) {**

**system("cls");**

**ifstream my\_input\_file;**

**my\_input\_file.open("instructions.txt");**

**if (!(my\_input\_file.is\_open()))**

**{**

**cout << "File cannot be opened.";**

**return 0;**

**}**

**char ch;**

**while (!my\_input\_file.eof())**

**{**

**my\_input\_file.get(ch); // using get() function**

**if (!my\_input\_file.eof())**

**cout << "\033[33m" << ch << "\033[0m";**

**}**

**cout << endl;**

**my\_input\_file.close();**

**cin >> ch;**

**}**

**else if (play\_game == 4) {**

**system("cls");**

**int score;**

**string record;**

**string nam;**

**fstream disp;**

**disp.open("Score.txt", ios::in);**

**while (!disp.eof())**

**{**

**getline(disp, record);**

**cout << endl << record;**

**}**

**cout << endl;**

**cin >> ch;**

**}**

**// 5 will exit the game simply by using exit keyword**

**else if (play\_game == 5)**

**{**

**system("cls");**

**cout << "\033[31m\t\t\t\t\t\t\tYou Want to Exit Game?\n\t\t\t\t\t\t\tY or y for yes\n\t\t\t\t\t\t\tb or B for NO\033[0m" << endl;**

**cin >> exi\_;**

**if (exi\_ == 'Y' || exi\_ == 'y') {**

**cout << "\033[33m\t\t\t\t\t\t\t You Exit Game!!\033[0m" << endl;**

**Sleep(2000);**

**exit(5);**

**}**

**else**

**cin >> ch;**

**}**

**cout << "\033[0m"; //to bold output on console**

**cin >> ch;**

**system("cls");**

**} while (ch == 'B' || ch == 'b');**

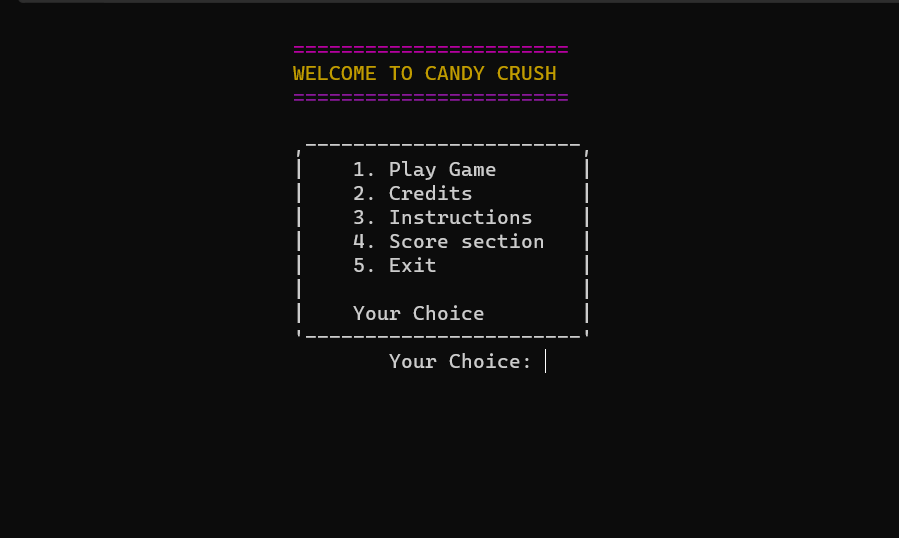
**system("pause");**

**return 0;**

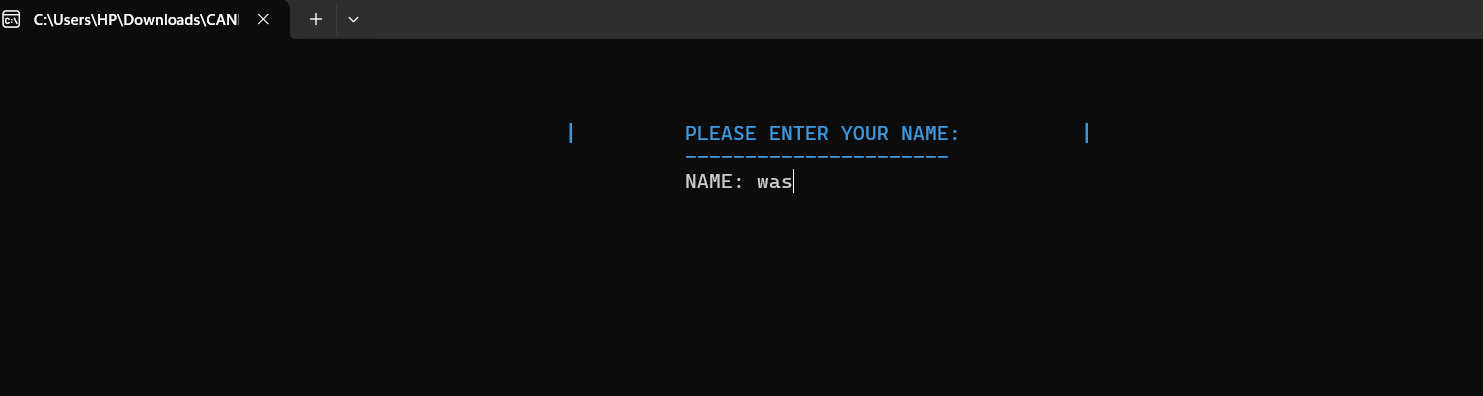
**}**

**🡨------SCREEN SHOTS------🡪**

**MAIN MENU:**

****

**After choosing 1 play game you need to enter the players name:**

****

**After Entering name choosing mode:**

**A screenshot of a computer

Description automatically generated**

**Hard Board :**

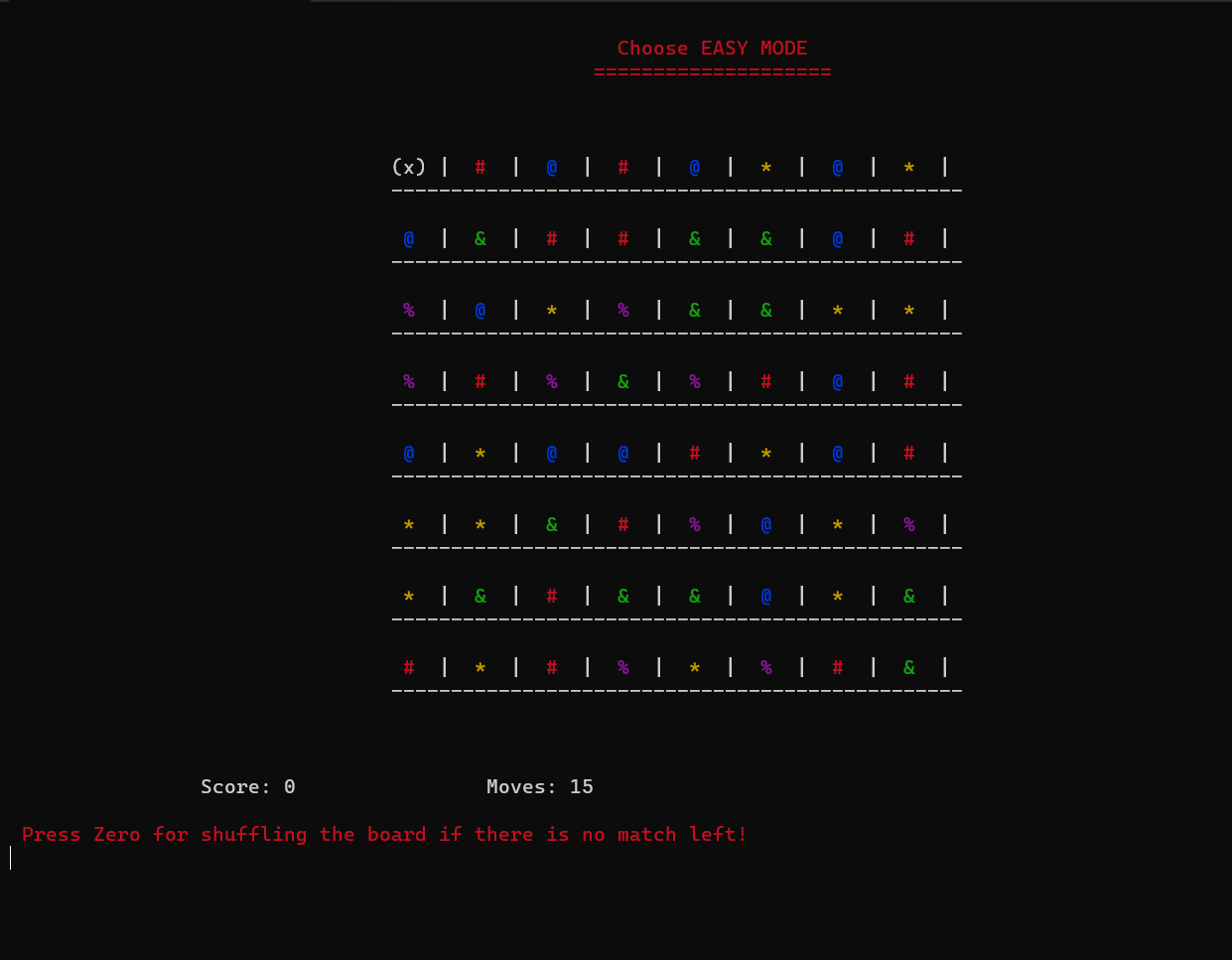
**You can see moves and score below the board and If u press zero the whole board gets shuffeled.Your name will be displayed on top on both board**

**A screenshot of a computer game

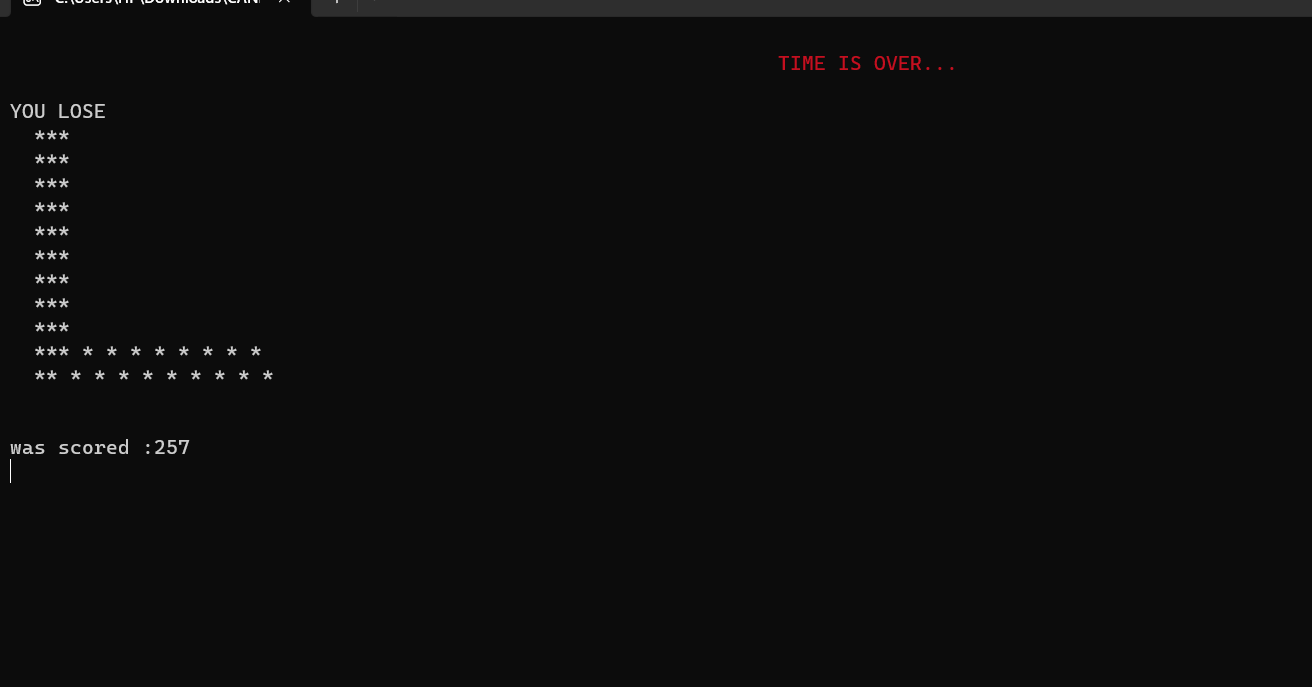
Description automatically generated**

**Easy Board:**

**You can see moves and score below the board and If u press zero the whole board gets shuffeled.Your name will be displayed on top on both boards.**

****

**After playing the game will display W for a Win (if score > 450)for L (if score<450 or timer runs out) For LOSE:**

****

**It will also display the score at the end.**

**A screenshot of a computer

Description automatically generated**

**The INSTRUCTIONS MENU:**

**A screen shot of a computer

Description automatically generated**

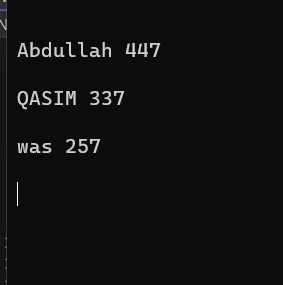
**The Credits Menu(File handling):**

**A screenshot of a computer

Description automatically generated**

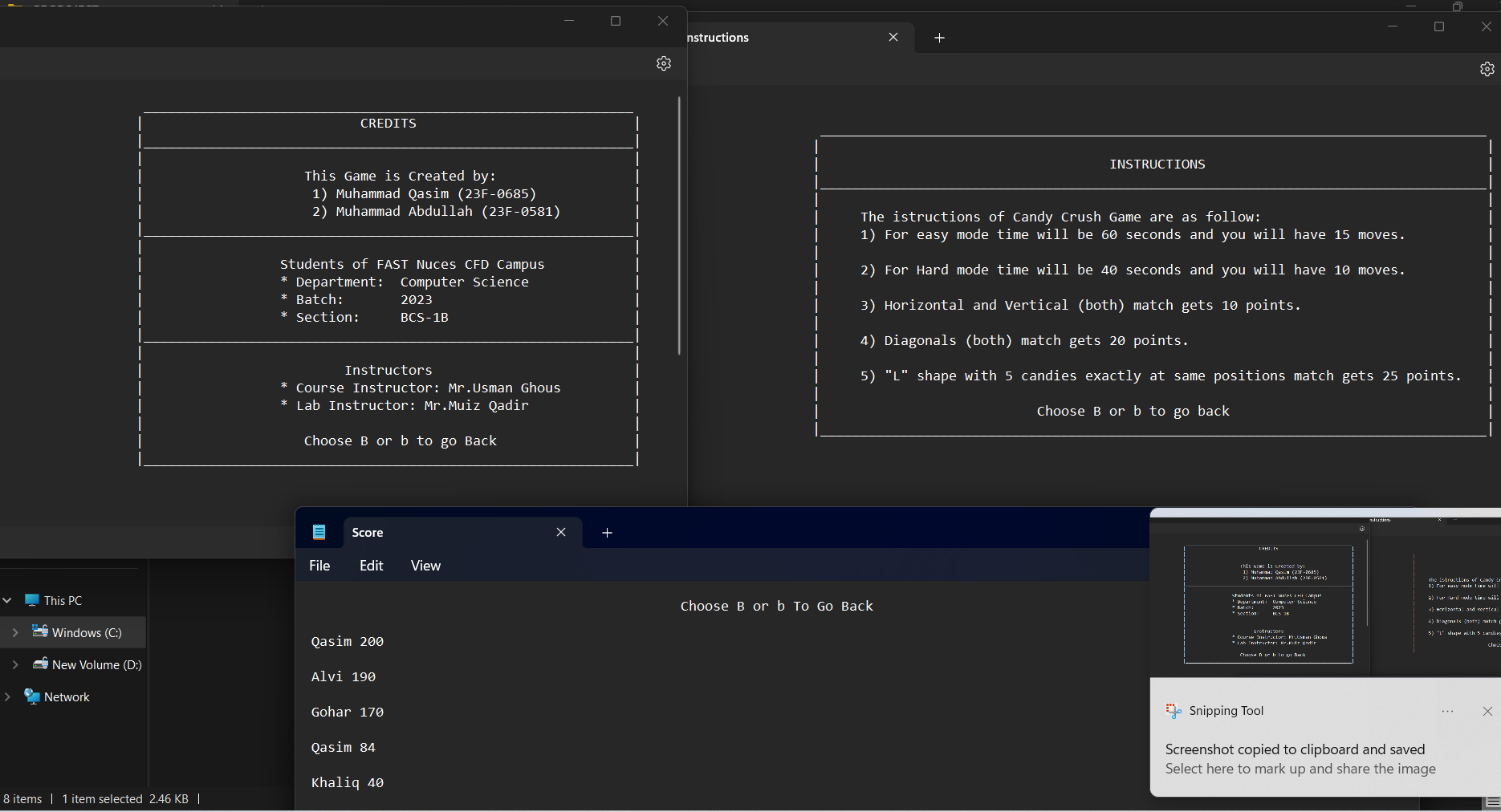
**ScoreCard Section:**

**Shows the score of top 3 and players who played the match.**

****

**MOREOVER YOU CAN COMEBACK TO THE MAIN MENU BY PRESSING B or b**

**TEXT FILES:**

****

**🡨---------HELPING REFERENCE--------🡪**

1. **Cplusplus.com**

* **For Timer**

1. **Windows.h**

* **Youtube**

1. **Keyswapping**

* **Sir + searching**