**CS-101 Project: Sci-Fi Planet Generator**

**Project Report: Planet Explorer and Alien Encounter MS Dos Style Interactive Experience**

**Authors: Yazdan Ali Khan (2024665), Raja Yawar Abbas (2024533), Hammad Shahid (2024389)**

**Link to repository:** [**Yhoqw/Planet\_gererator-CS-project**](https://github.com/Yhoqw/Planet_gererator-CS-project)

**1. Introduction**

The "Science Fiction Planet Generator" project is an interactive console-based application designed to engage users in exploring fictitious planets and their unique attributes. The program generates dynamic details about planets, their atmospheres, histories, and the alien races inhabiting them. Players also engage in mini-games to overcome challenges posed by alien inhabitants. This project demonstrates the use of random generation, seeding, Arrays, file I/O, and game logic in C++.

**2. Objectives**

1. Provide an engaging experience for users through the exploration of procedurally generated planets.
2. Demonstrate the application of key programming concepts such as:
   * Randomization and seeding
   * File handling (Saves and Loads planet information from a txt file)
   * Functions
   * Loops
   * IF/ELSE and Switch statements
   * User interaction
3. Develop mini-games to enhance the interactivity of the program.
4. Showcase teamwork and coding best practices.

**3. Features**

**Core Features**

1. **Planet Generation**
   * Randomly generates a planet name using pre-defined syllables and meshing together strings stored in an array
   * Assigns a random atmosphere and size from predefined lists.
   * Creates a unique history and events for each planet. While randomly choosing how many history events each planet will house
   * Generates Alien species for each planet and cycles number of races
2. **Alien Races**
   * Randomly generates alien races with unique appearances, technological levels, and forms of government.
3. **Mini-Games**
   * **Guessing Game:** Users must guess a randomly selected number within a limited number of attempts.
   * **Tic Tac Toe:** Users compete against an AI in a classic game of Tic Tac Toe.
4. **File Handling**
   * Allows users to save and load planet details to store previously visited planet.
5. **User Interface**
   * Interactive menu system
   * Title with cout statements

**4. Implementation**

**Technologies Used**

* **Language:** C++
* **Libraries:**
  + <iostream> for input and output.
  + <string> for handling textual data.
  + <ctime> for random number generation.
  + <fstream> for file handling.

**Code Structure**

The project adheres to a modular design to promote code clarity and reusability. Key components include:

1. **Constants and Data Arrays**
   * Predefined lists for atmospheres, sizes, histories, races, and alien details.
   * 2 dimensional arrays for tic tac toe game.

**Functions**

* + Each function handles a specific task, such as generating planet attributes, saving/loading files, or running mini-games.

1. **Main Program Logic**
   * A Do-While loop-based menu system guides the user through exploring new planets, loading saved planets, and exiting the program.

**5. Challenges Encountered**

1. **Randomization**
   * Ensuring that random generation did not repeat patterns too frequently.
   * Solution: Seeded the random number generator using the current time (srand(time(0))).
2. **Game Logic**
   * Developing AI logic for Tic Tac Toe that avoids obvious flaws.
   * Solution: Implemented a basic but effective AI that prioritizes available moves.
3. **File Handling**
   * Handling edge cases such as missing or corrupted files. (cerr)
   * Solution: Added error handling to ensure graceful failure with informative messages.
4. **Clearing text**
   * “system("cls");” clears the screen (only works on windows)

**6. Results**

The program successfully achieves its goals:

1. Generates unique and engaging planet profiles with detailed attributes.
2. Provides an interactive user experience through challenging and fun mini-games.
3. Implements a robust file-saving and loading mechanism to preserve user generation.
4. Making an early Dos RPG imitation
5. **Future Possibilities**

If we were ever to revisit the project or theoretically ponder what can or could have been added to the project.

1. **Fuel System:**
   * Add a fuel management system to simulate resource constraints during exploration and enhance gameplay loop.
2. **Graphical User Interface (GUI):**
   * Upgrade from a console-based interface to a GUI for improved aesthetics and usability.
3. **Enhanced Mini-Games:**
   * Introduce additional games or improve AI complexity in existing games.
4. **Expanded Planet Details:**
   * Include additional attributes such as gravity, temperature, and ecosystems.
5. **Porting the project on a dedicated Game engine**:
   * Rebuilding the project on an engine like unity or Godot to flesh out the project and make a playable build
6. **Actual utilization of the structures made in the program**

**8. Conclusion**

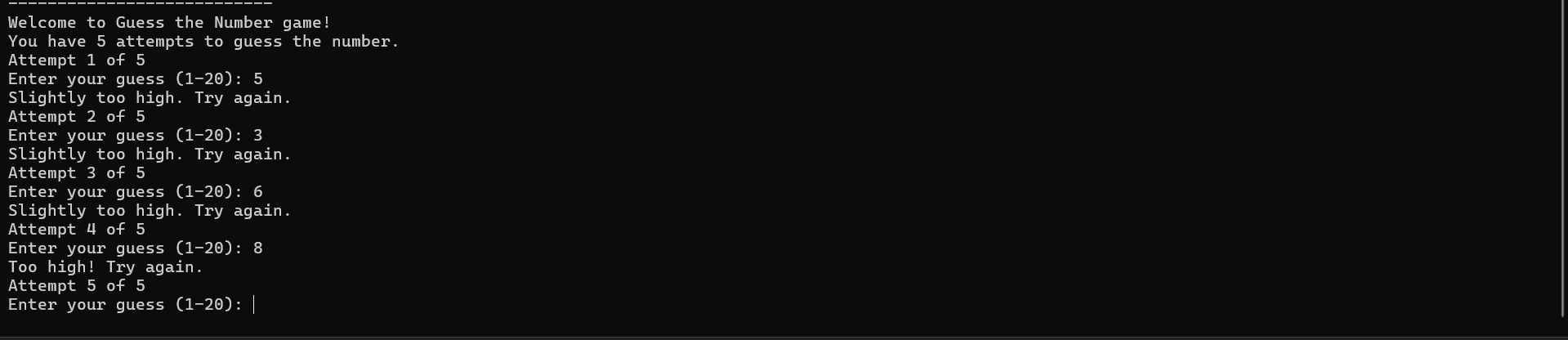
The "Science Fiction Planet Generator" project successfully combines creativity, programming skills, and user interactivity. It serves as a practical demonstration of C++ programming concepts. This project was an excellent opportunity for the team to apply theoretical knowledge in a hands-on environment, resulting in an engaging and functional application.

**9. References**

* C++ Standard Library Documentation
* Online tutorials
* Header files for <ctime>, <fstream>
* Various resources on file handling and random number generation in C++

**10. Output**

****





**11. Code:**

**#include <iostream>**

**#include <string>**

**#include <ctime>**

**#include <fstream>**

**//TO DO:**

**//fuel system**

**//structures**

**using namespace std;**

**// Constants**

**const int MAX\_NAME\_PARTS = 5;**

**const int MAX\_HISTORY\_EVENTS = 6;**

**const int MAX\_RACES = 6;**

**// Struct Definitions**

**struct Planet**

**{**

**string name;**

**string atmosphere;**

**string size;**

**};**

**struct Race**

**{**

**string name;**

**string appearance;**

**string techLevel;**

**string government;**

**};**

**// String arrays**

**string name\_parts[] = {"Zor", "Arg", "Tron", "Bel", "Nix", "Eld", "Omn", "Qua", "Tal", "Xyn", "Arra", "Kis"};**

**string atmospheres[] = {"Oxygen-rich", "Methane", "Carbon Dioxide", "Nitrogen", "Ammonia", "Helium"};**

**string size\_descriptions[] = {"Tiny", "Small", "Medium", "Large", "Massive"};**

**//string for planet history prompts (too many sci-fi references!!)**

**string histories[] =**

**{**

**"colonized by ancient explorers",**

**"destroyed in a cosmic war",**

**"thrived as a hub for trade",**

**"home to a great empire",**

**"witness to a catastrophic event",**

**"abandoned due to resource depletion",**

**"birthplace of a new religion",**

**"known for its unique flora and fauna",**

**"a target of space pirates",**

**"a secret research outpost",**

**"ravaged by nuclear war",**

**"survived the death star",**

**"probably one of starfields 1,692 planets",**

**"They claim that this is where they found out the answer to Life, the Universe and Everything",**

**"Great source of Spice",**

**"Home Planet to the Lisan al-Gaib?",**

**"Dr WHO was here",**

**"In proximity to a Halo ring",**

**"It has multiple moons",**

**};**

**//string for alien race names (I admit laser shark is kind of silly)**

**string races[] =**

**{**

**"Mandalorians", "Shanghili", "Xenomorphs", "Vulcans", "Geth", "Jedi", "Turians", "Daleks", "Martians", "Laser Sharks"**

**};**

**string appearances[] = {"Tall", "Short", "slender", "Reptilian", "Humanoid", "Insect-like"};**

**string techLevels[] = {"Primitive", "Industrial", "Spacefaring", "Advanced AI", "Interdimensional"};**

**string governments[] = {"Monarchy", "Republic", "Tribal council", "Technocracy", "Anarchy", "Theocracy"};**

**// Function prototypes**

**string generatePlanetName();**

**string generateAtmosphere();**

**string generateSize();**

**void generateHistory(string history[], int &eventCount);**

**void generateRaces(string raceList[], string raceDetails[], int &raceCount);**

**void displayPlanet(const string &name, const string &atmosphere, const string &size, const string history[], int eventCount, const string raceList[], const string raceDetails[], int raceCount);**

**void save\_planet\_file(const string &name, const string &atmosphere, const string &size, const string history[], int eventCount, const string raceList[], const string raceDetails[], int raceCount);**

**void load\_planet\_file();**

**int main()**

**{**

**//for seeding**

**srand(static\_cast<unsigned int>(time(0)));**

**int Input;**

**//Menu**

**cout << "=============================================================================================================================" << endl;**

**cout << " ###### # # # # ####### ####### ##### ####### # # ####### ###### # ####### ####### ###### " << endl;**

**cout << " # # # # # ## # # # # # # ## # # # # # # # # # # # " << endl;**

**cout << " ###### # # # # # # ##### # # #### ##### # # # ##### ###### # # # # # ###### " << endl;**

**cout << " # # ####### # # # # # # # # # # # # # # ####### # # # # # " << endl;**

**cout << " # # # # # ## # # # # # # ## # # # # # # # # # # " << endl;**

**cout << " # ####### # # # # ####### # ##### ####### # # ####### # # # # # ####### # # " << endl;**

**cout << "=============================================================================================================================" << endl;**

**cout << "A project by Yazdan Ali Khan, Raja Yawar Abbas and Hammad Shahid" << endl;**

**do**

**{**

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

**cout << "1. Explore a New Planet" << endl;**

**cout << "2. Explore previous Planet from File" << endl;**

**cout << "3. Exit Program" << endl;**

**cout << "Enter your choice: ";**

**cin >> Input;**

**switch (Input)**

**{**

**//case for planet Gereration**

**case 1:**

**{**

**string planetName = generatePlanetName();**

**string planetAtmosphere = generateAtmosphere();**

**string planetSize = generateSize();**

**// Generate history**

**string planetHistory[MAX\_HISTORY\_EVENTS];**

**int historyEventCount = 0;**

**generateHistory(planetHistory, historyEventCount);**

**//Generate Races**

**string planetRaces[MAX\_RACES];**

**string raceDetails[MAX\_RACES];**

**int raceCount = 0;**

**generateRaces(planetRaces, raceDetails, raceCount);**

**// Display and save the planet (using functions)**

**displayPlanet(planetName, planetAtmosphere, planetSize, planetHistory, historyEventCount, planetRaces, raceDetails, raceCount);**

**save\_planet\_file(planetName, planetAtmosphere, planetSize, planetHistory, historyEventCount, planetRaces, raceDetails, raceCount);**

**//Challenge logic**

**int Duel\_input;**

**cout << "\n" "---------------------------" << endl;**

**cout << "The inhabitants challenge you to a duel if you want to leave their planet. Press" "\n" "1. To play a guessing game " "\n" "2. To play a round of Tic tac toe" << endl;**

**cin >> Duel\_input;**

**switch (Duel\_input)**

**{**

**//Case for guessing game**

**case 1:**

**{**

**//Guessing game logic**

**int secret\_number = rand() % 20 + 1;**

**int max\_attempts = 5;**

**cout << "\n" "---------------------------" << endl;**

**cout << "Welcome to Guess the Number game!" << endl;**

**cout << "You have " << max\_attempts << " attempts to guess the number." << endl;**

**// Loop for the guessing game**

**for (int attempt = 1; attempt <= max\_attempts; attempt++)**

**{**

**cout << "Attempt " << attempt << " of " << max\_attempts << endl;**

**// Ask for the user's guess**

**int guess;**

**cout << "Enter your guess (1-20): ";**

**cin >> guess;**

**// Check if the guess is correct or not**

**if (guess > secret\_number)**

**{**

**//Feedback depending on how close the guess is for higher**

**if (guess > (secret\_number + 5) )**

**{**

**cout << "Too high! Try again." << endl;**

**}**

**else if (guess > secret\_number)**

**{**

**cout << "Slightly too high. Try again." << endl;**

**}**

**}**

**else if (guess < secret\_number)**

**{**

**//Feedback depending on how close the guess is for lower**

**if (guess < (secret\_number - 5) )**

**{**

**cout << "Too low! Try again." << endl;**

**}**

**else if (guess < secret\_number)**

**{**

**cout << "Slightly too low. Try again." << endl;**

**}**

**}**

**else**

**{**

**//to clear previous text and restart the loop**

**system("cls");**

**cout << "Congratulations! You guessed the right number!" << endl;**

**break;**

**}**

**// after attempts are finished give the answer**

**if (attempt == max\_attempts)**

**{**

**system("cls");**

**cout << "Sorry, you've used all your attempts. The correct number was " << secret\_number << "." << endl;**

**}**

**}**

**break;**

**}**

**//case for tic tac toe game**

**case 2:**

**{**

**char board[3][3] = {{'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}};**

**char player = 'X', computer = 'O';**

**cout << "Welcome to Tic Tac Toe!\n";**

**while (true)**

**{**

**// Display the board**

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

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**cout << board[i][j] << " ";**

**}**

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

**}**

**// Player's turn**

**int move;**

**cout << "Enter your move (1-9): ";**

**cin >> move;**

**int row = (move - 1) / 3, col = (move - 1) % 3;**

**if (move < 1 || move > 9 || board[row][col] == 'X' || board[row][col] == 'O')**

**{**

**cout << "Invalid move! Try again.\n";**

**continue;**

**}**

**board[row][col] = player;**

**// Check for player win**

**bool playerWin = false;**

**for (int i = 0; i < 3; i++)**

**{**

**//Horizontal and vertical**

**if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) || (board[0][i] == player && board[1][i] == player && board[2][i] == player))**

**{**

**playerWin = true;**

**break;**

**}**

**}**

**//diagonal**

**if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) || (board[0][2] == player && board[1][1] == player && board[2][0] == player))**

**{**

**playerWin = true;**

**}**

**if (playerWin)**

**{**

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

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**cout << board[i][j] << " ";**

**}**

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

**}**

**//clear previous text and display win statement**

**system("cls");**

**cout << "Congratulations! You win!\n";**

**break;**

**}**

**// Check if board is full (draw)**

**bool draw = true;**

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**if (board[i][j] != 'X' && board[i][j] != 'O')**

**{**

**draw = false;**

**}**

**}**

**}**

**if (draw)**

**{**

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

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**cout << board[i][j] << " ";**

**}**

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

**}**

**system("cls");**

**cout << "It's a draw!\n";**

**cout << endl;**

**break;**

**}**

**// Computer's turn**

**bool moveMade = false;**

**for (int i = 1; i <= 9 && !moveMade; i++)**

**{**

**row = (i - 1) / 3, col = (i - 1) % 3;**

**if (board[row][col] != 'X' && board[row][col] != 'O')**

**{**

**board[row][col] = computer;**

**moveMade = true;**

**}**

**}**

**// Check for computer win**

**bool computerWin = false;**

**for (int i = 0; i < 3; i++)**

**{**

**if ((board[i][0] == computer && board[i][1] == computer && board[i][2] == computer) || (board[0][i] == computer && board[1][i] == computer && board[2][i] == computer))**

**{**

**computerWin = true;**

**break;**

**}**

**}**

**if ((board[0][0] == computer && board[1][1] == computer && board[2][2] == computer) || (board[0][2] == computer && board[1][1] == computer && board[2][0] == computer))**

**{**

**computerWin = true;**

**}**

**if (computerWin)**

**{**

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

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**cout << board[i][j] << " ";**

**}**

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

**}**

**system("cls");**

**cout << "Computer wins! Better luck next time.\n";**

**break;**

**}**

**}**

**break;**

**}**

**}**

**//break for outer switch statement**

**break;**

**}**

**//case for loading saved planet**

**case 2:**

**{**

**load\_planet\_file();**

**break;**

**}**

**//case for exiting program**

**case 3:**

**{**

**cout << "Exiting program. Goodbye!" << endl;**

**break;**

**}**

**default:**

**{**

**//can use cerr here instead :) I checked**

**cout << "Invalid choice. Please try again." << endl;**

**}**

**}**

**} while (Input != 3);**

**return 0;**

**}**

**// Function to generate a random planet name**

**string generatePlanetName()**

**{**

**int parts = rand() % 3 + 2; // Between 2 and 4 parts**

**string name = "";**

**// Combining together multiple names to make one name essentially**

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

**{**

**name += name\_parts[rand() % MAX\_NAME\_PARTS];**

**}**

**return name;**

**}**

**// Function to generate a random atmosphere**

**string generateAtmosphere()**

**{**

**return atmospheres[rand() % (sizeof(atmospheres) / sizeof(atmospheres[0]))];**

**}**

**// Function to generate a random size**

**string generateSize()**

**{**

**return size\_descriptions[rand() % (sizeof(size\_descriptions) / sizeof(size\_descriptions[0]))];**

**}**

**// Function to generate a random history**

**void generateHistory(string history[], int &eventCount)**

**{**

**eventCount = rand() % MAX\_HISTORY\_EVENTS + 1; // Between 1 and MAX\_HISTORY\_EVENTS**

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

**{**

**history[i] = histories[rand() % (sizeof(histories) / sizeof(histories[0]))];**

**}**

**}**

**// Function to generate random races**

**void generateRaces(string raceList[], string raceDetails[], int &raceCount)**

**{**

**raceCount = rand() % MAX\_RACES + 1; // Between 1 and MAX\_RACES**

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

**{**

**raceList[i] = races[rand() % (sizeof(races) / sizeof(races[0]))];**

**//randmly generates (apearance, tech and government of alien races**

**raceDetails[i] = "Appearance { " + appearances[rand() % (sizeof(appearances) / sizeof(appearances[0]))] + " }, " +**

**"Tech Level { " + techLevels[rand() % (sizeof(techLevels) / sizeof(techLevels[0]))] + " }, " +**

**"Government { " + governments[rand() % (sizeof(governments) / sizeof(governments[0]))] + " }";**

**}**

**}**

**// Function to display the planet details**

**void displayPlanet(const string &name, const string &atmosphere, const string &size, const string history[], int eventCount, const string raceList[], const string raceDetails[], int raceCount)**

**{**

**cout << "\n" "Planet Details" << endl;**

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

**cout << "Name: " << name << endl;**

**cout << "Atmosphere: " << atmosphere << endl;**

**cout << "Size: " << size << endl;**

**cout << "\nHistory:" << endl;**

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

**{**

**cout << "- " << history[i] << endl;**

**}**

**cout << "\nRaces:" << endl;**

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

**{**

**cout << "- " << raceList[i] << ": " << raceDetails[i] << endl;**

**}**

**}**

**// Function to save the planet details to a file**

**void save\_planet\_file(const string &name, const string &atmosphere, const string &size, const string history[], int eventCount, const string raceList[], const string raceDetails[], int raceCount)**

**{**

**ofstream outFile("planet.txt");**

**if (outFile.is\_open())**

**{**

**outFile << name << endl;**

**outFile << atmosphere << endl;**

**outFile << size << endl;**

**outFile << eventCount << endl;**

**//for saving histories**

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

**{**

**outFile << history[i] << endl;**

**}**

**//for saving races**

**outFile << raceCount << endl;**

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

**{**

**outFile << raceList[i] << endl;**

**outFile << raceDetails[i] << endl;**

**}**

**//for closing the file after we are done with it**

**outFile.close();**

**cout << "\nPlanet saved to 'planet.txt' successfully." << endl;**

**}**

**else**

**{**

**cerr << "\nError: Unable to open file for writing." << endl;**

**}**

**}**

**// Function to load and display planet details from a file**

**void load\_planet\_file()**

**{**

**ifstream inFile("planet.txt");**

**if (inFile.is\_open())**

**{**

**string name, atmosphere, size;**

**int eventCount, raceCount;**

**getline(inFile, name);**

**getline(inFile, atmosphere);**

**getline(inFile, size);**

**inFile >> eventCount;**

**inFile.ignore(); // Ignore newline**

**//for loading histories**

**string history[MAX\_HISTORY\_EVENTS];**

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

**{**

**getline(inFile, history[i]);**

**}**

**inFile >> raceCount;**

**inFile.ignore(); // Ignore newline**

**//for loading races**

**string raceList[MAX\_RACES];**

**string raceDetails[MAX\_RACES];**

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

**{**

**getline(inFile, raceList[i]);**

**getline(inFile, raceDetails[i]);**

**}**

**//for closing the file after we are done with it**

**inFile.close();**

**// Display the loaded planet details**

**displayPlanet(name, atmosphere, size, history, eventCount, raceList, raceDetails, raceCount);**

**}**

**else**

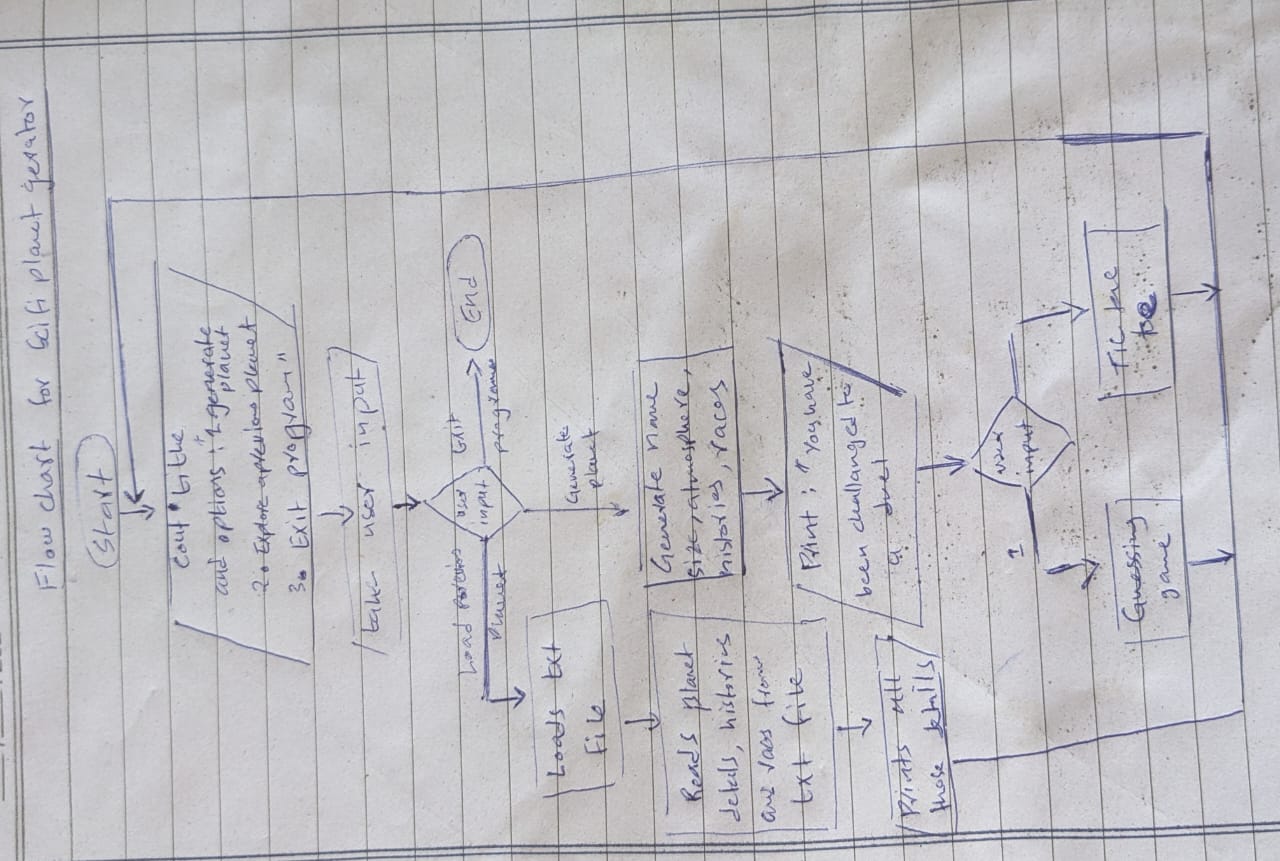
**{**

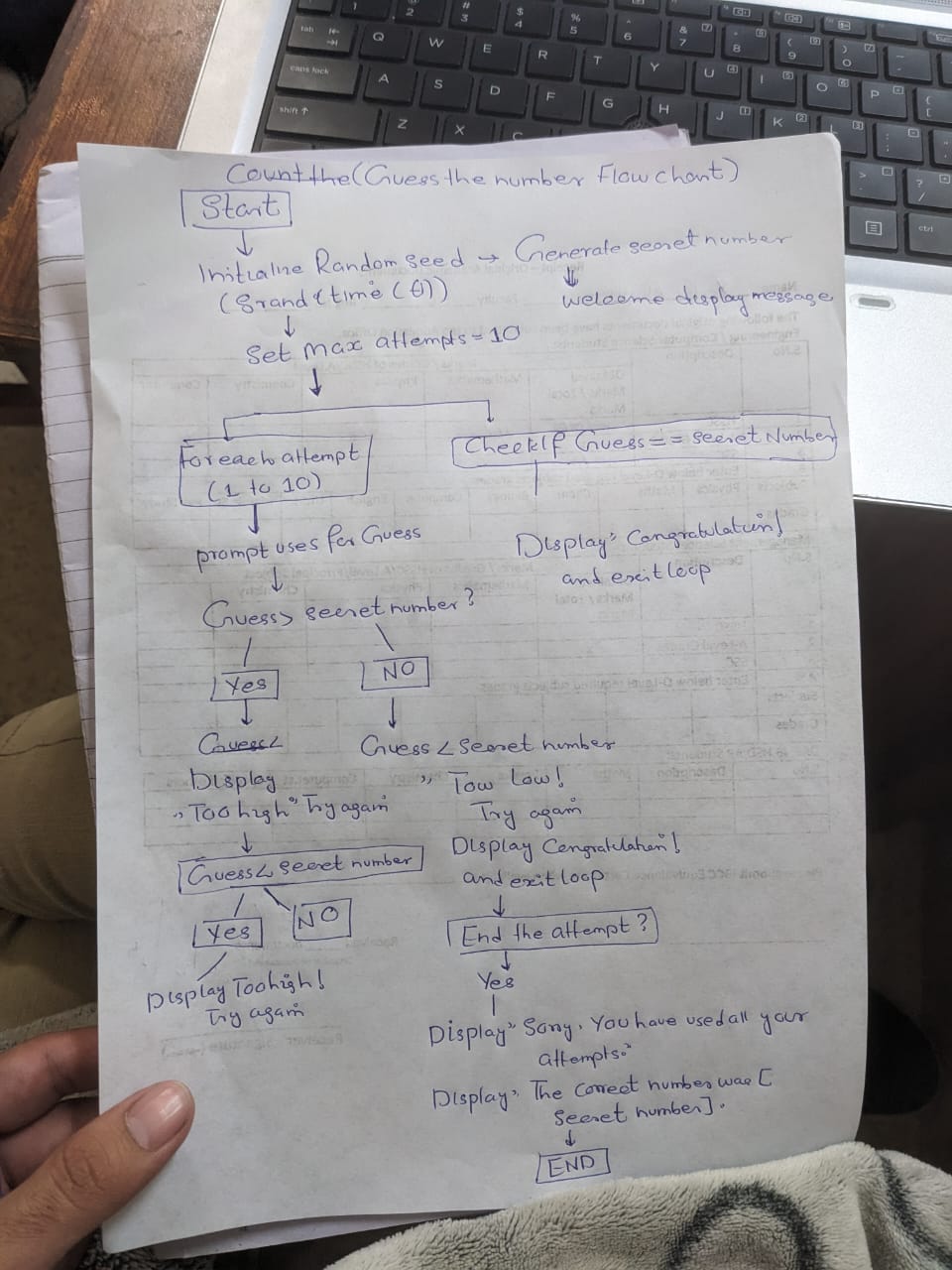
**cerr << "\nError: Unable to open file for reading." << endl;**

**}**

**}**

**Algorithms:**

****

****