



Interactive Guessing Game

**Egypt University of Informatics
Faculty of Engineering**

**Advanced Programming
(CSE121) Fall 2024**

Instructors:

Dr. Fatma Elzahraa Elshahaby

Eng. Shaymaa Elkaliouby

Table of Contents

1. Introduction	[Page 3]
o 1.1 Project Overview	[Page 3]
o 1.2 Project Scope	[Page 3]
o 1.3 Instructions to play	
2. System Requirements	[Page 4]
o 2.1 Hardware Requirements	[Page 4]
o 2.2 Software Requirements	[Page 4]
3. Implementation Details	[Page 5]
o 3.1 Game Design	[Page 5]
o 3.2 Dynamic Learning Mechanism	[Page 8]
o 3.3 File Handling	[Page 6]
4. Bonuses Features	[Page 9]
o 4.1 Multiple Modes	[Page 9]
o 4.2 Custom Mode Creation	[Page 9]
o 4.3 Game Timing	[Page 9]
o 4.4 Undo Functionality	[Page 9]
o 4.5 Multiplayer Mode	[Page 10]
5. Bonuses Implementation	[Page 10]
o 5.1 Mode Selection	[Page 10]
o 5.2 Custom Mode Creation	[Page 10]
o 5.3 Game Timing	[Page 10]
o 5.4 Undo Functionality	[Page 10]
o 5.5 Multiplayer Mode	[Page 10]
6. Challenges and Solutions	[Page 11]
o 6.1 File Handling Errors	[Page 11]
o 6.2 Incorrect Tree Traversal	[Page 11]
7. GUI Description	[Page 12]
8. Testing the code	[Page 25]
9. Conclusion	[Page 30]
10. references	[Page 31]
11. team members	[Page 35]

Objective:

The goal is to design an interactive game that predicts an outcome (e.g., a course, club, or country) based on a series of user-provided answers. The game incorporates a dynamic learning mechanism, allowing it to adapt and improve by saving new information from each session.

Introduction:

This project explores the creation of an interactive guessing game using C programming, focusing on the prediction of academic courses based on user responses. The game uses a binary tree data structure to represent a decision-making process, where each node corresponds to a question or potential answer.

The user interacts with the game by answering yes/no questions related to courses, such as their topics, and the time taking in.

If the game makes an incorrect guess, it learns from the user by adding new information to its binary tree structure. This approach enables the program to evolve over time, becoming more accurate with each interaction.

By employing file handling techniques, the game retains its learned data across sessions, making it a persistent and continually improving system.

System Requirements

To successfully develop and run the interactive guessing game, the following system requirements must be met:

1. Hardware Requirements

- **Processor:** Minimum 1 GHz (Recommended: 2 GHz or faster)
- **RAM:** At least 4 GB (Recommended: 8 GB for smoother execution)
- **Storage:** 10 MB of free disk space for the program and saved data
- **Display:** A monitor capable of displaying text output

2. Software Requirements

- **Operating System:**
 - Windows 7 or later
 - macOS 10.12 or later
 - Any modern Linux distribution (e.g., Ubuntu 18.04+)
- **C Compiler:**
 - GCC (GNU Compiler Collection) version 7.0 or later
 - Clang version 5.0 or later
 - Any compiler compatible with the C99 standard or later
- **Editor/IDE:**
 - Visual Studio Code, Code: Blocks, or any text editor with C support

Instructions for playing

1. Starting the Game:

- The game begins with a menu where you can select from Solo or Multiplayer mode.
- You'll be asked a series of Yes/No questions to determine your selection.

2. Menu Options:

- **Option 1:** Course-related guessing game.
- **Option 2:** Country-related guessing game.
- **Option 3:** Programming language guessing game.
- **Option 4:** Football club guessing game.
- **Option 5:** Car-related guessing game.
- **Option 6:** Create your own custom guessing game.

3. Answering Questions:

- At each step, answer **Yes (y)** or **No (n)** to the displayed question.

4. Undo Feature:

- If you make a mistake, you can use the Undo option to revert to the previous question.

5. Custom Games:

- Use the "Create your own game" option to design a new guessing tree. You'll be prompted to provide questions and guesses.

6. Timer Feature:

- The game tracks how long each turn takes and displays it at the end of your turn.

7. Tree Display:

- At any point, you can print the decision tree to visualize the game's structure.

Technical Notes

We uploaded it as game.c and scratch.exe (we tried to make it game.exe and C-lion)

Our header file is “5leha_3la_allah”

Our implementation files is (play_game.c \\ mode and construction.c \\extra.c)

- **External Terminal in C-Lion:**

- This program uses a system (" cls ") to clear the screen after every menu or question. To ensure this works in C-Lion:
 1. Go to **Settings > Build, Execution, Deployment > Console.**
 2. Select **Use external terminal for running applications.**
 3. Apply the changes and rerun the program.
- This ensures the game interface remains clean and organized.

- **File Paths:**

- Update the file paths in the source code to match your system.
- Example:

```
// Returns the path to a specific game file based on the user's choice
string path_choose(int number_selection) {
    string course_guess="D:\\programming\\Uni courses\\2nd year\\Advanced prog\\Game guess final\\course_guess.txt";
    string country_guess="D:\\programming\\Uni courses\\2nd year\\Advanced prog\\Game guess final\\country_guess.txt";
    string club_guess="D:\\programming\\Uni courses\\2nd year\\Advanced prog\\Game guess final\\club_guess.txt";
    string car_guess="D:\\programming\\Uni courses\\2nd year\\Advanced prog\\Game guess final\\car_guess.txt";
    string programming_guess="D:\\programming\\Uni courses\\2nd year\\Advanced prog\\Game guess final\\programming_guess.txt";
    string build_new_tree = "D:\\programming\\Uni courses\\2nd year\\Advanced prog\\Game guess final\\new_tree.txt";
```

How to Exit the Game

- You can exit the game anytime by selecting the **Quit** option from the menu.
- Use the "Save Changes" feature to preserve any updates you've made to the decision tree.

Implementation Details

The interactive guessing game is implemented using the C programming language. It leverages a binary tree data structure to represent and traverse a decision-making process. This section outlines the core components of the implementation, including game design, learning mechanism, file handling, and key programming concepts.

1. Game Design

The game functions as a decision tree, where:

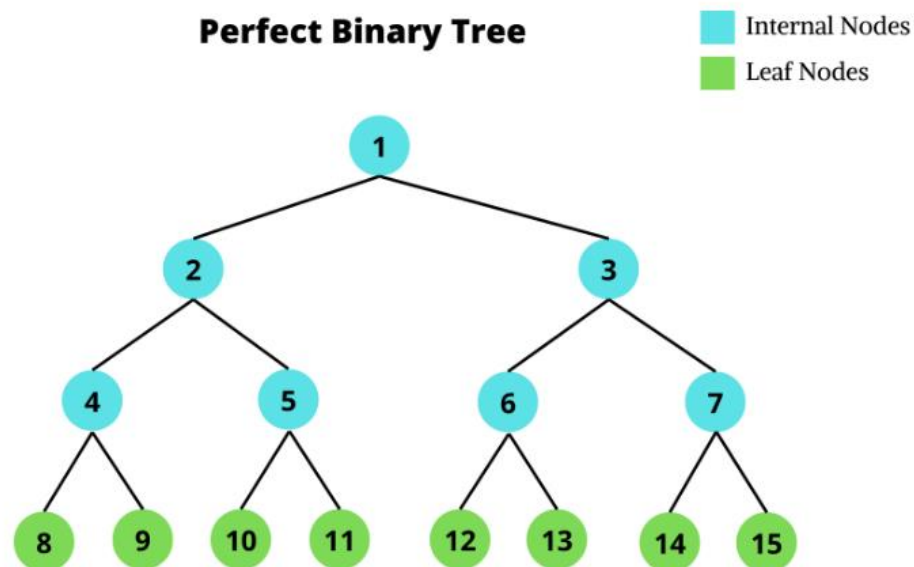
- **Nodes:** Represent questions or answers.
- **Branches:** Represent yes/no responses to questions.

The game starts at the root node, asking a question. Based on the user's response (pre-order traversal):

- If **yes**: the program traverses to the left child node.
- If **no**: the program traverses to the right child node.

When a leaf node is reached:

- The game makes a guess about the course.
- If the guess is correct, the game terminates.
- If incorrect, the game learns from the user by asking for the correct answer and a distinguishing question.



2. Dynamic Learning Mechanism

If the game makes an incorrect guess, it:

1. Prompts the user to provide the correct course name.
 2. Requests a distinguishing question to differentiate the guessed course from the correct course.
 3. Updates the binary tree by:
 - Converting the current leaf node into a question node.
 - Creating two child nodes:
 1. One for the correct course.
 2. One for the previously guessed course.
-

3. File Handling

To preserve knowledge across sessions, the game saves and loads the binary tree from a file. The process involves:

- **Saving Data:**
 - The tree is serialized using pre-order traversal.
 - Each node is written on the file, tagged as a question or a course.
- **Loading Data:**
 - The file is deserialized to reconstruct the binary tree at program start.
 - A recursive function is used to create nodes and link them appropriately.

Bonuses features

- **Multiple Modes:**

- The game will support multiple modes, including:
 - Course
 - Car
 - Country
 - Club
 - Programming Language
- Each mode will have its own independent binary tree data structure and save data file.

- **Custom Mode Creation:**

- Users will have the option to create custom modes.
- In custom mode creation:
 - The user will manually build the initial binary tree structure by providing questions and answers.
 - This allows for personalized and customized gameplay experiences.

- **Game Timing:**

- Each game session will be timed.
- The game will start a timer when a mode is selected.
- The timer will stop when the game makes a correct guess.
- The final time will be displayed to the user.

- **Undo Functionality:**

- Users will be able to undo their previous answer during a game session.
- This provides flexibility and allows for corrections if the user makes a mistake.

- **Multiplayer Mode:**

- The game will include a multiplayer mode.
- Two or more players can participate in a single game session.
- Players will take turns answering questions.
- If the game makes an incorrect guess, the player whose turn it was gains a point.
- The player with the highest score at the end of the game wins.

Bonuses implementation

- **Mode Selection:**

- Implement a menu or interface for selecting the desired game mode.
- Load the appropriate binary tree and game data for the selected mode.

- **Custom Mode Creation:**

- Implement a user-friendly interface for creating custom modes.
- Provide clear instructions and guidance for building the initial binary tree.

- **Game Timing:**

- Use a suitable timing mechanism to track the game duration.
- Display the final time in a user-friendly format in seconds

- **Undo Functionality:**

- Implement a mechanism to store the user's answer history.
- Allow the user to "undo" the last answer and revert the game state.

- **Multiplayer Mode:**

- Implement player tracking and scoring mechanisms.
- Handle player turns and inputs appropriately.
- Display the current score and game status to all players.

Challenges and Solutions

1. File Handling Errors

- **Challenge:** Errors occurred when saving or loading the binary tree from the file, especially due to incorrect formatting or file permission issues.
- **Solution:**
 - Used explicit error checking after file operations (e.g., checking `fopen()` or `fscanf()` return values).
 - Added default data as a fallback if the file could not be loaded.
 - Test for the directory.

2. Incorrect Tree Traversal

- **Challenge:** The game crashed or gave incorrect results due to null pointer dereferencing during tree traversal.
- **Solution:**
 - Added validation checks before accessing node pointers.
 - Implemented robust testing of tree traversal logic with edge cases (e.g., empty trees, missing child nodes).

3. Handling Multiple Modes

- **Challenge:** Ensuring correct loading, saving, and switching between different game modes (Course, Car, Country, etc.) while maintaining data integrity.
- **Solution:**
 - Implemented separate data files for each game mode.
 - Thoroughly tested mode switching functionality to prevent data corruption or unexpected behavior.

4. Custom Mode Creation

- **Challenge:** Ensuring the user-created binary tree structure is valid and free from errors that could cause crashes or unexpected game behavior.
- **Solution:**
 - Implemented input validation to prevent invalid user input during custom mode creation.
 - Provided clear error messages and guidance to the user to help them create valid custom modes.
 - Thoroughly tested custom modes to identify and address any potential issues.

5. Multiplayer Mode

- **Challenge:** Synchronizing player turns, handling player input correctly, and ensuring fair and accurate scoring in the multiplayer mode.
- **Solution:**
 - Implemented robust turn-based logic to ensure proper player order and prevent cheating.
 - Thoroughly tested multiplayer functionality with multiple players to identify and resolve any issues.

6. Undo Functionality

- **Challenge:** Implementing the undo functionality correctly without causing unexpected behavior or data corruption.
- **Solution:**
 - Implemented a robust undo mechanism that accurately tracks and reverts game state changes.
 - Thoroughly tested the undo functionality in various scenarios to ensure it works as expected.

7. Serialization and deserialization

- **Challenge:** there are no references to this part or video in c language
- **Solution:**
 - Search on geeks for geeks and other programming websites and uses Chat GPT to understand the concept.

Some explain for external screen usage:

We use C-lion to write and we clear screen, so we use the external terminal in C-lion.

And the command for clearing

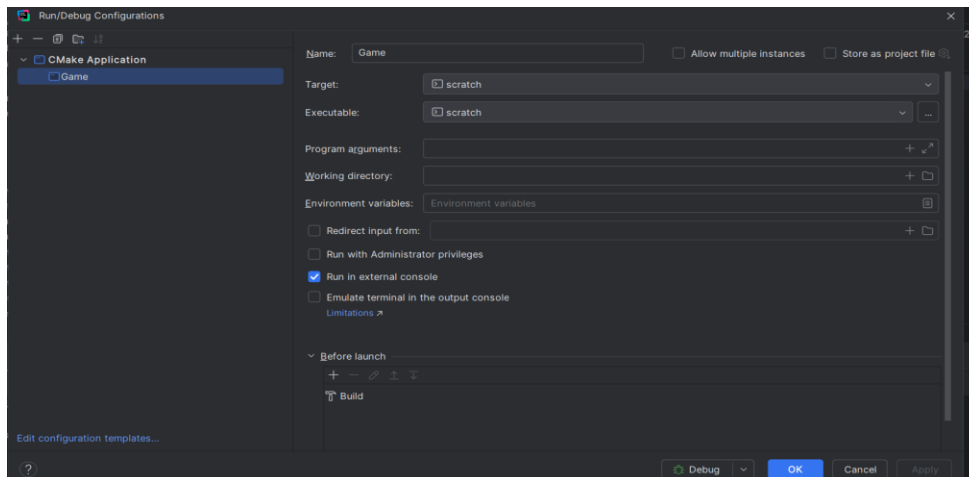
```
▼ cls
+ :: In C, system("cls") is used to clear the console (or terminal) screen on Windows operating systems. It calls the system shell to execute the cls command, which is the command for clearing the screen in Windows command prompt.

#include <stdlib.h>

int main() {
    // Clear the screen (only works on Windows)
    system("cls");
}

#include <stdlib.h>
int main() {
    // Clear the screen (only works on Windows)
    system("cls");
    return 0;
}
```

To activate this



GUI DOCUMENTATION

```
#include "raylib.h"
#include "stdio.h"
#define RAYGUI_IMPLEMENTATION
#include "raygui.h"
#include "5leha 3la allah.h"

typedef enum {SELECTWALLPAPER,MAIN,GAME,SETTINGS,CREDITS,GAMEOVER}screen;
int num_start_menu;
int num_quit_menu;
char ans1[10]="null";
int myconstruction=1;
int answeyes=0;
int numyes=0;
int numno=0;
int stats=0;
int answerno=0;
int submit=0;
char question[100];
char ans_yes[100];
char ans_no[100];

nodptr ngui=NULL;
nodptr prev=NULL;
```

In this part, we included all the libraries we used in the process.

We also initialized some important global variables that's needs to be used in the implementation file of our header file.

In the first part of our main, we load all the music, backgrounds, and textures files we need, along with initializing the GUI window, and some variables that's to be used soon in the main.

Target FPS is set to 144, and game music is played here.

GUI style and text size of the whole game are initialized.

```
int count_questions = 0;
InitWindow(1800,900,"GUESS ME");
InitAudioDevice();
screen x = SELECTWALLPAPER;
float volume_music = 0.5f;

int selectedWallpaper = 0;
Music gamemusic = LoadMusicStream("D:\\Engineering study\\programming\\GUI\\game_music.mp3");
Music gamemusic1 = LoadMusicStream("D:\\Engineering study\\programming\\GUI\\car_music.mp3");
Music gamemusic2 = LoadMusicStream("D:\\Engineering study\\programming\\GUI\\club_music.mp3");
Music gamemusic3 = LoadMusicStream("D:\\Engineering study\\programming\\GUI\\programming_music.mp3");
Music gamemusic4 = LoadMusicStream("D:\\Engineering study\\programming\\GUI\\country_music.mp3");
Texture2D background = LoadTexture("D:\\Engineering study\\programming\\GUI\\course.jpg");//course
Texture2D texture2 = LoadTexture("D:\\Engineering study\\programming\\GUI\\flags1.jpg");
Texture2D texture3 = LoadTexture("D:\\Engineering study\\programming\\GUI\\prog.jpg");//programming
Texture2D texture4 = LoadTexture("D:\\Engineering study\\programming\\GUI\\club.jpg");//clubs
Texture2D texture5 = LoadTexture("D:\\Engineering study\\programming\\GUI\\car5.jpg");
Texture2D no_react = LoadTexture("D:\\Engineering study\\programming\\GUI\\no_react.jpg");
Texture2D gameover = LoadTexture("D:\\Engineering study\\programming\\GUI\\gameover1.jpg");
Texture2D logo = LoadTexture("D:\\Engineering study\\programming\\GUI\\logo.jpg");
Texture2D emoji = LoadTexture("D:\\Engineering study\\programming\\GUI\\ava.jpg");

SetTargetFPS(144);
PlayMusicStream(gamemusic);
GuiSetStyle(DEFAULT, TEXT_SIZE, 20);
```

```

while (!WindowShouldClose()) {
    if (IsKeyPressed(KEY_ESCAPE)) {
        CloseWindow(); // Close the window when ESC is pressed
    }

    UpdateMusicStream(gamemusic);
    UpdateMusicStream(gamemusic1);
    UpdateMusicStream(gamemusic2);
    UpdateMusicStream(gamemusic3);
    UpdateMusicStream(gamemusic4);

    switch(selectedWallpaper){
        case 1:
            ClearBackground(BLACK);
            break;
        case 2:
            ClearBackground(BLACK);
            break;
        case 3:
            ClearBackground(BLACK);
            break;
        case 4:
            ClearBackground(PURPLE);
            break;
        case 5:
            ClearBackground(BLACK);
            break;
    }
}

```

Everything appearing on the window, every sound playing in the background, and anything happening while the window is open usually happen here in this while loop.

In the part we start by making the escape key close the game window, then we update all the game music files (making it start from the beginning if the music file ends), then we select a clear background color based on selectedWallpaper's value.

Inside the while loop, everything is updated instantly, so we treated the code inside it as if it's all running in the same time. That's why the code is full of if-else statements, as we put all the code blocks inside conditional statements to make specific parts run while other parts remain untouched.

Category/Wallpaper Selection Screen:

If the enum variable x is equal to “SELECTWALLPAPER”, we are now in the category/wallpaper selection screen.

In this screen, we have the logo of the game along with 5 buttons, each button changes the value of “selectedWallpaper”, gamemusic, and the enum variable x depending on what theme does the button indicate.

Changing the x’s value to “MAIN” makes the condition of the if-statement in the image false, which closes the category/wallpaper selection screen to open the main screen of the game.

```
if(x==SELECTWALLPAPER) {
    ClearBackground(BLACK);
    DrawTexture(logo, 640, 00, WHITE);
    DrawText("Welcome to Guess Game ",550,400,60,WHITE);
    DrawText("Please choose the mode you want to play ",320,600,25,WHITE);
    if (GuiButton((Rectangle) {300, 650, 250, 50}, "Course guess")) {
        selectedWallpaper = 1;
        x=MAIN;
        PlayMusicStream(gamemusic);
        StopMusicStream(gamemusic1);
        StopMusicStream(gamemusic2);
        StopMusicStream(gamemusic3);
        StopMusicStream(gamemusic4);
    }
    else if (GuiButton((Rectangle) {550, 650, 250, 50}, "Country guess")) {
        {
            selectedWallpaper = 2;
            x=MAIN;
            PlayMusicStream(gamemusic4);
            StopMusicStream(gamemusic1);
            StopMusicStream(gamemusic2);
            StopMusicStream(gamemusic3);
            StopMusicStream(gamemusic);
        }
    }
    else if (GuiButton((Rectangle) {800, 650, 250, 50}, "Programming guess")) {
        {
            selectedWallpaper = 3;
            x=MAIN;
            PlayMusicStream(gamemusic3);
            StopMusicStream(gamemusic1);
            StopMusicStream(gamemusic2);
            StopMusicStream(gamemusic);
            StopMusicStream(gamemusic4);
        }
    }
    else if (GuiButton((Rectangle) {1050, 650, 250, 50}, "Club guess")) {
        {
            selectedWallpaper = 4;
            PlayMusicStream(gamemusic2);
            StopMusicStream(gamemusic1);
            StopMusicStream(gamemusic);
            StopMusicStream(gamemusic3);
            StopMusicStream(gamemusic4);
            x=MAIN;
        }
    }
    else if (GuiButton((Rectangle) {1300, 650, 250, 50}, "Car guess")) {
        {
            selectedWallpaper = 5;
            PlayMusicStream(gamemusic1);
            StopMusicStream(gamemusic2);
            StopMusicStream(gamemusic);
            StopMusicStream(gamemusic3);
            StopMusicStream(gamemusic4);
            x=MAIN;
        }
    }
}
```


Main Screen:

```
BeginDrawing();
if (x == MAIN){
    switch (selectedWallpaper) {
        case 1:
            DrawTexture(background, 1000, 00, WHITE); // Draw background 1
            if(myconstruction==1) {
                mode_choose(1);
                ngui=root;
                myconstruction=0;
            }
            break;
        case 2:
            DrawTexture(texture2, 0, 0, WHITE); // Draw background 2
            if(myconstruction==1) {
                mode_choose(2);
                ngui=root;
                myconstruction=0;
            }
            break;
        case 3:
            DrawTexture(texture3, 25, 0, WHITE); // Draw background 3
            if(myconstruction==1) {
                mode_choose(3);
                ngui=root;
                myconstruction=0;
            }
            break;
        case 4:
            DrawTexture(texture4, 0, 00, WHITE); // Draw background 3
            if(myconstruction==1) {
                mode_choose(4);
                ngui=root;
                myconstruction=0;
            }
            break;
        case 5:
            DrawTexture(texture5, 1000, 0, WHITE); // Draw background
            if(myconstruction==1) {
                mode_choose(5);
                ngui=root;
                myconstruction=0;
            }
            break;
    }
}
```

The main screen, along with the rest of the screens, are all inside the `BeginDrawing();` `EndDrawing();` block of raylib.

Everything that's to be drawn on the window should be inside this block.

In our main screen, we start by drawing the background of the chosen theme from the previous screen, assigning the root pointer of the tree to our tree-navigation pointer “ngui”, and making sure there is no problems with the drawn background or the tree-navigation pointer by assigning the variable “myconstruction” to be 0.

Secondly, a text is drawn on the screen that indicates what category/theme is chosen.

Lastly, 4 buttons are made, each button changes the current screen to another screen.

The first button changes the x's value to be "GAME" which starts the game screen. It also assigns 1 to "num_start_menu" which is a global variable needed in the implementation file for the start_menu function.

The second button changes the x's value to be "GAMEOVER" which basically ends the game. It also assigns 0 to the global variable num_quit_menu that's needed in the implementation file for the quit menu function, and 1 to the num_start_menu.

The third button goes to the settings screen, and the last button goes to the credits one.

```
switch(selectedWallpaper){
    case 1:
        DrawText("Welcome to course guess",550,300,45,WHITE);
        break;
    case 2:
        DrawText("Welcome to country guess",550,300,45,WHITE);
        break;
    case 3:
        DrawText("Welcome to programming guess",550,300,45,WHITE);
        break;
    case 4:
        DrawText("Welcome to club guess",550,300,45,WHITE);
        break;
    case 5:
        DrawText("Welcome to car guess",550,300,45,WHITE);
        break;
}
if (GuiButton((Rectangle){500,475,200,50},"START")){
    x = GAME;
    num_start_menu = 1;
}
else if (GuiButton((Rectangle){700,475,200,50},"Quit")){
    x = GAMEOVER;
    num_quit_menu = 0;
    num_start_menu =1;
}
else if (GuiButton((Rectangle){900,475,200,50},"Settings")){
    x = SETTINGS;
}
else if (GuiButton((Rectangle){1100,475,200,50},"Credits")){
    x = CREDITS;
}
}
```

Settings Screen:

```
else if (x == SETTINGS){
    UnloadTexture(logo);
    switch (selectedWallpaper) {
        case 1:
            DrawTexture(background, 1000, 00, WHITE); // Draw background 1
            break;
        case 2:
            DrawTexture(texture2, 0, 0, WHITE); // Draw background 2
            break;
        case 3:
            DrawTexture(texture3, 25, 0, WHITE); // Draw background 3
            break;
        case 4:
            DrawTexture(texture4, 0, 00, WHITE); // Draw background 4
            break;
        case 5:
            DrawTexture(texture5, 1000, 0, WHITE); // Draw background 5
            break;
    }
    if (GuiButton((Rectangle){10,10,100,50},"Back")) {
        x = MAIN;
    }
    GuiLabel((Rectangle){250,160,600,200}, TextFormat("music volume :%.1f",volume_music));
    if (GuiSlider((Rectangle){250,300,300,25},"Volume",NULL,&volume_music,0.0,1.5)){
        SetMusicVolume(gamemusic,volume_music);
        SetMusicVolume(gamemusic1,volume_music);
        SetMusicVolume(gamemusic2,volume_music);
        SetMusicVolume(gamemusic3,volume_music);
        SetMusicVolume(gamemusic4,volume_music);
    }
}
```

We start by drawing the background for the selected theme, and unloading the logo texture.

Then a back button is made, which changes the current screen to the main screen.

A slider is made to control the volume of the current music in play.

There is a label above the slider that indicates the current music volume.

Credits Screen:

```
else if (x == CREDITS){
    UnloadTexture(logo);
    switch (selectedWallpaper) {
        case 1:
            DrawTexture(background, 1000, 00, WHITE); // Draw background 1
            break;
        case 2:
            DrawTexture(texture2, 0, 0, WHITE); // Draw background 2
            break;
        case 3:
            DrawTexture(texture3, 25, 0, WHITE); // Draw background 3
            break;
        case 4:
            DrawTexture(texture4, 0, 00, WHITE); // Draw background 4
            break;
        case 5:
            DrawTexture(texture5, 1000, 0, WHITE); // Draw background 5
            break;
    }
    if (GuiButton((Rectangle){10,10,100,50},"Back")){
        x = MAIN;
    }
    DrawText("Credits", (GetScreenWidth()- MeasureText("Credits", 50))/2, 200, 50, WHITE);
    DrawText("GUI:", 10, 300, 30, PURPLE);
    DrawText("Saif Haytham", 10, 340, 30, WHITE);
    DrawText("23-101282", 400, 340, 30, WHITE);
    DrawText("Ziad Mohamed", 10, 380, 30, WHITE);
    DrawText("23-101280", 400, 380, 30, WHITE);

    DrawText("Code:", 10, 450, 30, PURPLE);
    DrawText("3abood e1 ahbl", 10, 490, 30, WHITE);
    DrawText("23-101279", 400, 490, 30, WHITE);
    DrawText("Moataz Hindy", 10, 530, 30, WHITE);
    DrawText("23-101290", 400, 530, 30, WHITE);
    DrawText("Hassan Ahmed", 10, 570, 30, WHITE);
    DrawText("Blank", 400, 570, 30, WHITE);

    DrawText("Data Base:", 10, 640, 30, PURPLE);
    DrawText("Ahmed Zaytoun", 10, 680, 30, WHITE);
    DrawText("23-101329", 400, 680, 30, WHITE);
}
EndDrawing();
```

We start by drawing the background for the selected theme, and unloading the logo texture.

Then a back button is made, which changes the current screen to the main screen.

Then we draw our team's names and IDs.

Then we end drawing using the function `EndDrawing()`;

Note:

The upcoming Game and Game Over screens are both inside the drawing block.

Gameover Screen:

The GAMEOVER state is triggered when the player completes all questions in the game. It serves as the end screen and provides the following functionalities:

1. **Reset Stats:** All game-related stats, such as the number of questions answered, "Yes" and "No" answers, and player statistics, are reset to zero.
2. **Game Over Screen:** A game over image is displayed. If the player answered "Yes" at any point, a congratulatory message appears.
3. **Restart Option:** A button labelled "Restart new mode" is shown. When clicked, it resets the game to the initial state, stops any current background music, and starts new music. This state provides a clear end to the game while allowing the player to restart the game with a fresh setup and can choose new mode to play.
4. **Restart Option:** A button labelled "Restart same mode" is shown. When clicked, it resets the game to the initial state, so that the user can start playing the same mode once again.
5. **Quit option:** A button labelled "Quit" is shown. When clicked, it ends the game window.

```
else if(x==GAMEOVER){
    ClearBackground(LIGHTGRAY);
    DrawTexture(gameover, 00, -200, WHITE);
    if(answeryes) DrawText("Congratulations ",550,100,100,WHITE);
    if (GuiButton((Rectangle){500,650,200,50},"Restart new mode")){
        myconstruction=1;
        PlayMusicStream(gamemusic);
        StopMusicStream(gamemusic1);
        StopMusicStream(gamemusic2);
        StopMusicStream(gamemusic3);
        StopMusicStream(gamemusic4);
        ngui=root;
        x=SELECTWALLPAPER;
    }
    if (GuiButton((Rectangle){750,650,200,50},"Restart same mode")){
        ngui=root;
        x=MAIN;
    }
    if (GuiButton((Rectangle){1000,650,200,50},"save updates")){
        FILE *f = fopen(path, "w");
        update_tree_file(f,root);
        fclose(f);
    }
    if (GuiButton((Rectangle){1250,650,200,50},"Quit")){
        CloseWindow();
    }
}
```

Game Screen:

The game begins by displaying a background and a question prompt, depending on the value of selectedWallpaper.

The player is presented with a "Yes" or "No" button to answer the current question.

If the player answers "Yes", the game moves to the next question (or asks for a new one if the current question is the last).

If the player answers "No", the game navigates to a different branch or adds new questions if needed.

```
else if (x==GAME) {  
    UnloadTexture(logo);  
    switch (selectedWallpaper) {  
        case 1:  
            DrawTexture(background, 1000, 00, WHITE); // Draw background 1  
            break;  
        case 2:  
            DrawTexture(texture2, 0, 0, WHITE); // Draw background 2  
            break;  
        case 3:  
            DrawTexture(texture3, 25, 0, WHITE); // Draw background 3  
            break;  
        case 4:  
            DrawTexture(texture4, 0, 00, WHITE); // Draw background 4  
            break;  
        case 5:  
            DrawTexture(texture5, 1000, 0, WHITE); // Draw background 5  
            break;  
    }  
}
```

The question is displayed on the screen, and the prompt changes depending on whether the question is a simple guess or if it's part of the decision tree.

ngui->q holds the current question text, which is drawn at specific coordinates (150, 50).

If there are no further questions (ngui->yes == NULL && ngui->no == NULL), the game prompts the user to add a new question.

"Yes" Button: When clicked, the game records a "Yes" answer, moves to the next question, or adds a new question to the tree.

"No" Button: When clicked, the game records a "No" answer and proceeds similarly to the "Yes" button.

```
if (GuiButton((Rectangle){1200,200,200,50},"no")){
    decision = 2;
    answerno=1;
    if( ngui->yes==NULL && ngui->no==NULL ){
        answerno=2;
    }
    answeyes=0;
}
if(answerno){
    DrawTexture(emoji, 10, 300, WHITE);
    DrawText("HMMMMMMMMMM! ",220,400,25,WHITE);
}
```

```
int decision;
if(ngui->yes==NULL && ngui->no==NULL && !(answerno==2) ){
    DrawText("I guess: ",10,50,30,WHITE);
}
else{
    DrawText("Q: ",100,50,30,WHITE);
}
DrawText(ngui->q,150,50,35,WHITE);
if (GuiButton((Rectangle){1000,200,200,50},"Yes")){
    decision = 1;
    answeyes=1;
    answerno=0;
}
if(!(answeyes || answerno)){
    DrawTexture(emoji, 10, 300, WHITE);
    DrawText("Hi my friend let's start playing ",220,400,25,WHITE);
}
if(answeyes){
    if(!(ngui->yes==NULL && ngui->no==NULL) ){
        DrawTexture(emoji, 10, 300, WHITE);
        DrawText("Well....The next question is ",220,400,25,WHITE);
    }
    else{
        DrawTexture(emoji, 10, 300, WHITE);
        DrawText("please tell me that i guessed correct",220,400,25,WHITE);
    }
}
}
if (decision==1) {
    numyes++;
    strcpy(ans1, "yes");
    if(ngui->yes==NULL && ngui->no==NULL ){
        x=GAMEOVER;
    }
    if(!(ngui->yes==NULL && ngui->no==NULL) ){
        ngui=ngui->yes;
    }
    decision=0;
}
```

If there are no questions left (`ngui->yes == NULL && ngui->no == NULL`), the player is prompted to input a new question.

The new question, along with the possible answers for "Yes" and "No", is added to the decision tree.

If the decision tree reaches a leaf node (i.e., there are no further questions), the game is considered over. The player may then add new questions to expand the tree or restart the game.

The game tracks:

Total number of questions in the tree
(`count_questions`). Number of questions
answered with "Yes" (`numyes`).

Number of questions answered with "No"
(`numno`).

These statistics are displayed upon the player's
request.

Drawing Text and Buttons:

DrawText: Used to render text on the screen.
Text includes questions, messages, and stats.

GuiButton: Draws buttons and handles
interactions with them. It checks if the player
clicks the "Yes", "No", or other buttons.

```
if( ngui->yes==NULL && ngui->no==NULL ){
    if( answerno==2) {
        DrawText("let's add some information ",400,400,25,WHITE);
        DrawText("what question do you want to add to distinguish between answers?", 220, 500, 20,WHITE);
        if (!submit) {
            GuiTextBox((Rectangle) {210, 550, 200, 30}, question, 100, true);
        } else {
            GuiTextBox((Rectangle) {210, 550, 200, 30}, question, 100, false);
        }
        if (GuiButton((Rectangle) {380, 550, 200, 50}, "submit")) {
            strcpy(ngui->q, question);
            submit = 1;
        }
        if (submit == 1) {
            DrawText("what answer in case of yes?", 120, 600, 20, WHITE);
            if (submit == 1) {
                GuiTextBox((Rectangle) {210, 650, 200, 30}, ans_yes, 100, true);
            } else {
                GuiTextBox((Rectangle) {210, 650, 200, 30}, ans_yes, 100, false);
            }
            if (GuiButton((Rectangle) {380, 650, 200, 50}, "submit")) {
                submit = 2;
            }
        }
        if (submit == 2) {
            DrawText("what answer in case of no?", 120, 700, 20, WHITE);
            GuiTextBox((Rectangle) {210, 750, 200, 30}, ans_no, 100, true);
            if (GuiButton((Rectangle) {380, 750, 200, 50}, "submit")) {
                decsion(ngui);
                submit=0;
                x = GAMEOVER;
            }
        }
    }
}
```


Navigating the Decision Tree:

The decision tree is navigated based on the player's answers.

When the player selects "Yes" or "No", the ngui pointer updates to the next question node (ngui = ngui->yes or ngui = ngui->no).

If the player reaches a node with no further options (ngui->yes == NULL && ngui->no == NULL), the game prompts for a new question to be added to the tree.

Undo Button: The player can undo the last decision, moving back to the previous question in the decision tree.

"Print Tree" Button: If enabled, this button shows a graphical representation of the decision tree.

"Player Stats" Button: This button shows statistics of the player's progress, including the number of questions answered and the breakdown of "Yes" and "No" answers.

```
if(decision==2){
    numno++;
    strcpy(ans1,"no");
    if(!(ngui->yes==NULL && ngui->no==NULL) ){
        ngui=ngui->no;
    }
    decision=0;
}
if (GuiButton((Rectangle){1400,200,200,50},"undo")){
    decision = 3;
}
if (decision==3) {
    if(answeryes){
        numyes--;
    }
    if(answerno){
        numno--;
    }
    strcpy(ans1, "u");
    ngui=searchTree(root, ngui->q);
    decision=0;
}
if (GuiButton((Rectangle){1400,250,200,50},"print tree")){
    decision = 4;
}
if (decision==4) {
    DrawTree(root, 900, 300, 1450 / 2.5, 40);
}
if (GuiButton((Rectangle){1000,250,200,50},"player stats")){
    stats= 1;
}
if (GuiButton((Rectangle){1200,250,200,50},"Turn off stats")){
    stats= 0;
}
if (stats==1) {
    // Set font size to 30
    // Set label color (affects all labels)
    // GuiSetStyle(DEFAULT, GUI_TEXT_COLOR, RED);
    DrawText("your stats till now are :",50,630,20,WHITE);
    GuiLabel((Rectangle){50,650,600,200}, TextFormat("number of questions is :%.1d",count_questions));
    GuiLabel((Rectangle){50,700,600,200}, TextFormat("number of questions answered with yes :%.1d",numyes));
    GuiLabel((Rectangle){50,750,600,200}, TextFormat("number of questions answered with no :%.1d",numno));
}
```

Resources:

- <https://www.raylib.com/cheatsheet/cheatsheet.html>
- <https://www.raylib.com/examples.html>
- <https://github.com/raysan5/raygui/tree/master/examples>

And chat gpt :)

Testing the code

The game will be rigorously tested to ensure its functionality, accuracy, and robustness.

- **Unit Tests:** Individual components (binary tree operations, file I/O functions) will be tested with various inputs to verify their correctness.
- **Integration Tests:** The entire game flow will be tested to ensure that all components work together as expected. This includes:
 - Correctness of predictions.
 - Effective learning and updating of the binary tree.
 - Proper handling of user input and game states.
 - Successful loading and saving of game data.
 - Thorough testing of each game mode, including custom modes.
 - Testing of the timing and undo functionality.
 - Extensive testing of the multiplayer mode, including edge cases and scenarios with multiple players.
- **User Acceptance Testing (UAT):** A small group of users will play the game and provide feedback on its usability, playability, and overall experience.
- **Edge Case Testing:** Specific scenarios will be tested to ensure the game handles unexpected situations gracefully, such as invalid user input, empty data files, or unexpected errors during file operations.

Snippets from the game

<pre>E:\Empty\bin\Debug\Empty.e × + ▾ ----- Hello! Welcome to our guess game ----- Enter a number to 0 quit 1 play solo 2 play multi mode</pre>	<pre>----- Select the next game thematic ----- 1.Course 2.Country 3.Programming language 4.Foot Club 5.Car 6.Create ur Guess game</pre>
<pre>----- Make a guess in ur mind and we will know it ----- Rules of input: "yes" for YES "no" for NO "undo" for return Back Let's play ----- Is this Year 1? :</pre>	<pre>----- Guess is Innovation1. Is The guess correct? </pre>
<pre>----- Guess is Innovation1. Is The guess correct? yes Got it! ----- The mode lasted for 437.00 seconds. ----- Enter a number to 0 quit 1 print tree 2 Restart same mode in solo 3 restart with solo 4 restart with multi</pre>	<pre>Final Scores: AZaytoun: 0 points AGomaa: 2 points Deuce! ----- The mode lasted for 211.00 seconds. ----- Enter a number to 0 quit 1 print tree 2 Restart same mode in solo 3 restart with solo 4 restart with multi</pre>
<pre>----- Rules: The Winner is who defeat the game wins one point for each victory ----- Enter the number of players: 2</pre>	<pre>----- Rules of input: "yes" for YES "no" for NO ----- player[1]: AZaytoun Round 1 ----- is it a sports car ?</pre>

```
E:\Empty\bin\Debug\Empty.e X + v

Rules:
The Winner is who defeat the game wins
one point for each victory

Enter the number of players: 2
Enter the name of player 1: AZaytoun
Enter the name of player 2: AGomaa

Guess is China.
Is The guess correct?
no

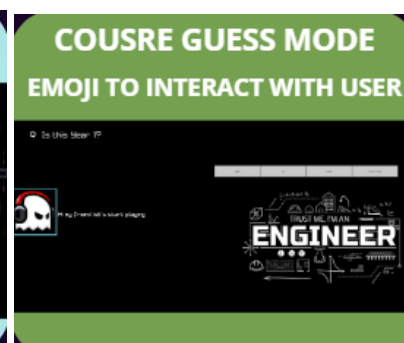
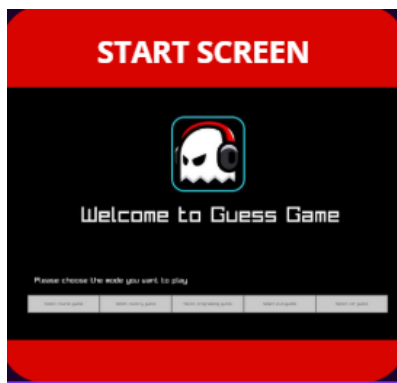
Please enter a question to distinguish this Guess:

Is This country is known for its small size?
What is the Answer if yes?
Brunai
What is the Answer if no?
Mongolia
Do you want to save changes for next times ? yes/no
yes
The changes were saved

The mode lasted for 107.00 seconds.

Enter a number to
0 quit
1 print tree
2 Restart same mode in solo
3 restart with solo
4 restart with multi
```

```
-----
Guess is China.
Is The guess correct?
no
-----
Please enter a question to distinguish this Guess:
-----
Is This country is known for its small size?
What is the Answer if yes?
```



Conclusion

This project successfully demonstrates the implementation of an interactive guessing game utilizing a binary tree data structure in the C programming language. The game effectively predicts outcomes based on user input and incorporates a dynamic learning mechanism to enhance its accuracy over time.

Key features such as multiple game modes, custom mode creation, game timing, undo functionality, and a multiplayer mode significantly enrich the gameplay experience. Through rigorous testing and careful attention to detail, the game has been developed to be robust, user-friendly, and engaging.

This project serves as valuable learning experience in:

- Structures
- linked lists
- binary tree
- recursion
- binary recursion
- dealing with files
- others

(all with c programming language). It showcases the practical application of binary trees and demonstrates the potential for creating interactive and intelligent systems that can adapt and learn from user interactions.

References

We used YouTube and global websites to learn the algorithm to write the code

<https://youtu.be/q529aPtmm24?si=wILO7eJoSNqJcuzn>

<https://youtu.be/U1fvq1rAcgY?si=5Vp3MidGnPG6vR4K>

https://youtu.be/PXtosFYXbdo?si=9ojB8-m_Wo-ADPat

<https://stackoverflow.com/questions/16099271/including-files-in-a-header-vs-implementation-file>

<https://youtu.be/YJip4KI8le4?si=GkviZToGD6mixsxp>

<https://www.geeksforgeeks.org/tree-data-structure/>

<https://www.geeksforgeeks.org/binary-tree-data-structure/>

<https://www.geeksforgeeks.org/preorder-traversal-of-binary-tree/>

<https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>

<https://www.geeksforgeeks.org/implement-a-stack-using-singly-linked-list/>

<https://www.youtube.com/watch?v=dqgZd8qvXoY>

LEC 9 : [https://drive.google.com/drive/folders/1Zth52iuX5TaW4TqJvd9tJWj-rOX3k4qd?usp=drive link](https://drive.google.com/drive/folders/1Zth52iuX5TaW4TqJvd9tJWj-rOX3k4qd?usp=drive_link)

<https://www.youtube.com/watch?v=pcKY4hjDrxk&t=969s&pp=ygUNYWJkdWxiYXJpIGRmcw%3D%3D>

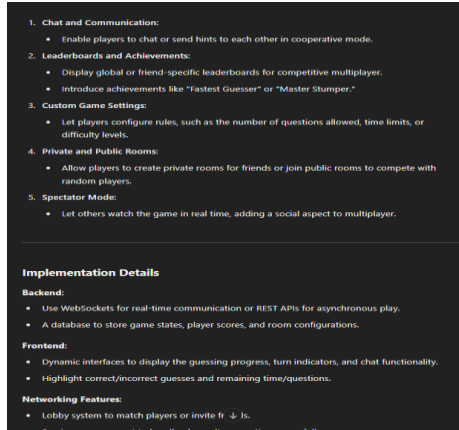
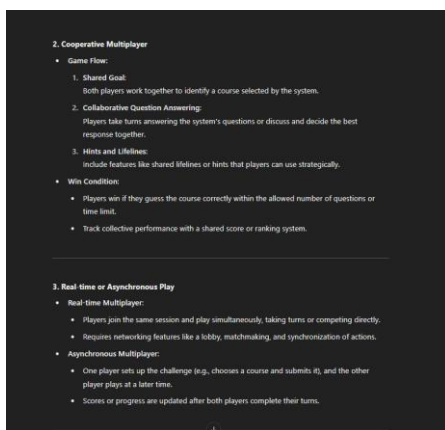
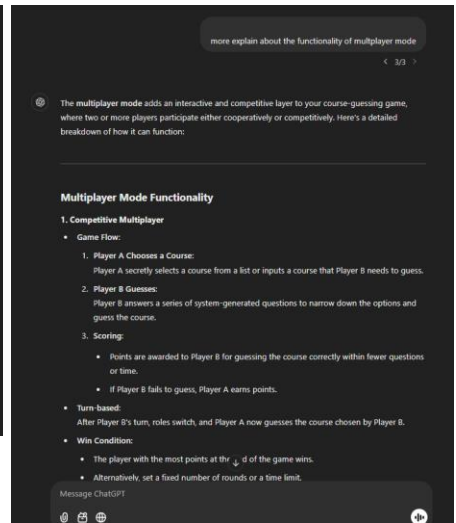
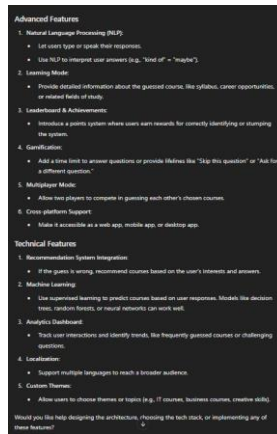
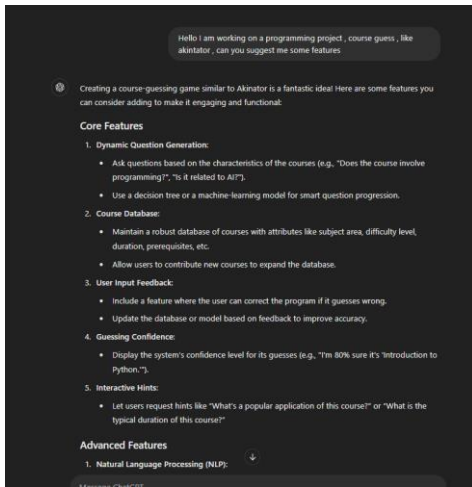
https://youtu.be/xo41NfT8218?si=ejxs_R1jFVLjB1yJ

https://youtu.be/cf_TpRNWlxc?si=gCcxKgplZW18epuY

<https://www.geeksforgeeks.org/c-exit-abort-and-assert-functions/>

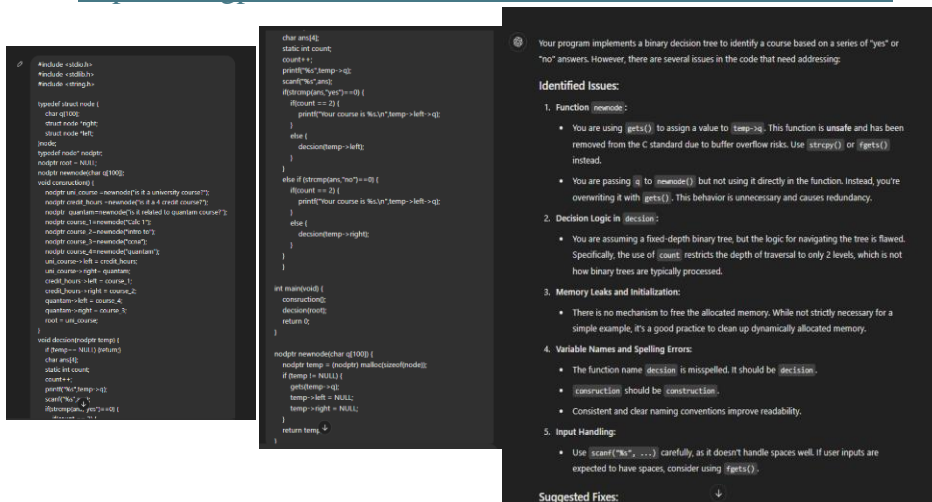
We also Used Chat GPT for new functions and looking for features:

<https://chatgpt.com/c/676156a4-6e70-8004-9b3a-83c069d8f782>



This at the beginning of our learning of binary trees and implementation of prototype of project, we used it to check our steps of learning from mentioned resources

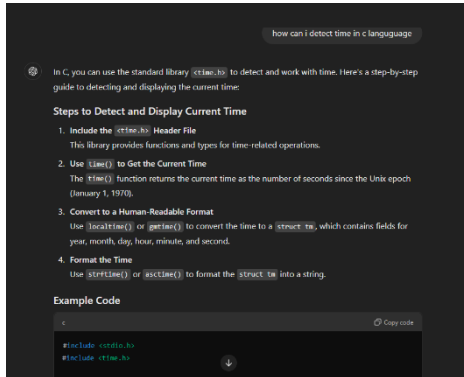
<https://chatgpt.com/c/67529e6e-2764-8004-adf6-33fb58e79740>



For the rest of the chat click the link

This our search for time logic in c

<https://chatgpt.com/c/676ef266-6950-8012-844f-b2d806d4dd50>



how can i detect time in c language

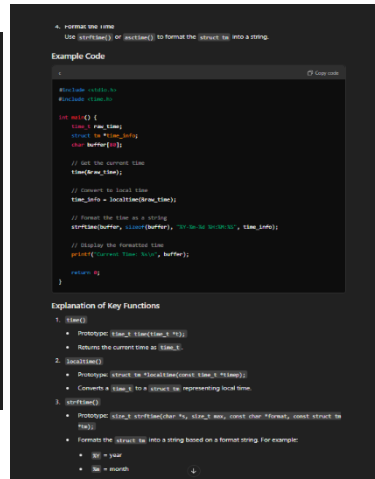
In C, you can use the standard library `<time.h>` to detect and work with time. Here's a step-by-step guide to detecting and displaying the current time:

Steps to Detect and Display Current Time

- Include the `<time.h>` Header File**
This library provides functions and types for time-related operations.
- Use `time()` to Get the Current Time**
The `time()` function returns the current time as the number of seconds since the Unix epoch (January 1, 1970).
- Convert to a Human-Readable Format**
Use `localtime()` or `gmtime()` to convert the time to a `struct tm`, which contains fields for year, month, day, hour, minute, and second.
- Format the Time**
Use `strftime()` or `asctime()` to format the `struct tm` into a string.

Example Code

```
c
#include <stdio.h>
#include <time.h>
```



```
4. Format the time
Use strftime() or asctime() to format the struct tm into a string.

Example Code

#include <stdio.h>
#include <time.h>

int main() {
    time_t raw_time;
    struct tm *time_info;
    char buffer[80];

    // Get the current time
    time(&raw_time);

    // Convert to local time
    time_info = localtime(&raw_time);

    // Format the time as a string
    strftime(buffer, sizeof(buffer), "%d-%m-%Y %H:%M:%S", time_info);

    // Display the formatted time
    printf("Current Time: %s", buffer);

    return 0;
}
```

Explanation of Key Functions

- time()**
 - Prototype: `time_t time(time_t *t)`
 - Returns the current time as `time_t`.
- localtime()**
 - Prototype: `struct tm *localtime(const time_t *t)`
 - Converts a `time_t` to a `struct tm`, representing local time.
- strftime()**
 - Prototype: `int strftime(char *s, rsize_t n, const char *format, const struct tm *tm)`
 - Formats the `struct tm` into a string based on a format string. For example:
 - `%d` = day
 - `%m` = month
 - `%Y` = year

<https://www.geeksforgeeks.org/time-function-in-c/>

We also used some files from last year class (file of country and clubs) and did some manipulation to fit our program.

Feature of external terminal, I learnt from my friend.

Team:

Name	ID
Hassan Ahmed	23-101330
Ahmed Zaytoun	23-101329
Moataz Ashraf	23-101290
Seif Haytham	23-101282
Ziad Moahmed	23-101281
Abdulrahman Gomaa	23-101279

Team Roles/Responsibilities

Responsible Team Member(s)	Role
Moataz Ashraf	Undo feature (using stack), Learning process of the program and updating the tree, poster, programing language file making
Ziad Mohammed	creating the cmake and header for the GUI-creating the settings screen and main screen -printing the file tree- participated in linking GUI with the code , Poster
Hassan Ahmed	Construction of trees from file and update the file (File Handling), Create new guess game feature, Report documentation, Courses file making
Ahmed Zaytoun	Players' stats in GUI, Multiplayer mode, Documentation 1, Timer
Seif Haytham	Link GUI with code-Themes and music-Interactive emoji-Traversing in questions, answers, screens and modes-updating the tree - Presentation - GUI video - car file making
Abdulrahman Gomaa	<p>1/ Overall supervision over the program</p> <p>2/play the game from start to make the guess</p> <p>3/video editing</p>