Design & Media College

An Assignment Work submitted for the subject

Interactive Media Development

By

Jashwanth SR

University Registration No.: 238348730010

2023 – 2024

ALAGAPPA UNIVERSITY

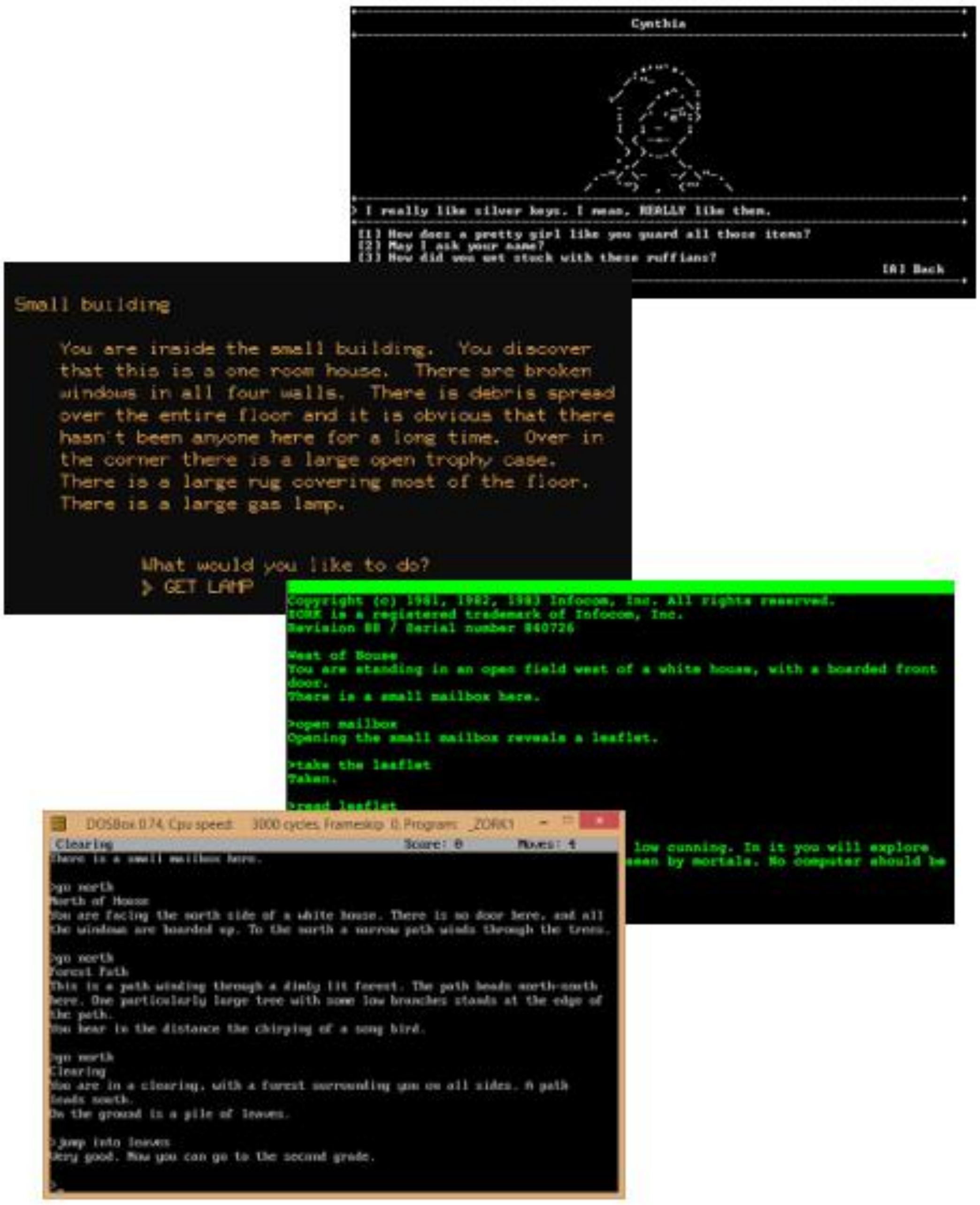
(A State University Established by the Government of Tamil Nadu in 1985, Accredited with A+ Grade by NAAC (CGPA 3.64) in the Third Cycle, Graded as Category-I University and Granted Autonomy by MHRD-UGC, MHRD-NIRF 2020 Rank : 36, QS 2020 India Rank : 24)

KARAIKUDI - 630 003, Tamil Nadu, India

INDEX

Brief:

The project involves creating a text-based game using the C++ programming language. In this game, players interact with the game solely through text input and output in the console. The gameplay revolves around presenting the player with choices, and based on their decisions, the game progresses with different outcomes or scenarios. The objective is to implement a compelling and interactive experience using only text-based interfaces and C++ logic.



```
Cynthia

> I really like silver keys. I mean, REALLY like them.

[1] How does a pretty girl like you guard all those items?
[2] May I ask your name?
[3] How did you get stuck with these ruffians?
                                                [A] Back
```

```
Small building

    You are inside the small building.  You discover
    that this is a one room house.  There are broken
    windows in all four walls.  There is debris spread
    over the entire floor and it is obvious that there
    hasn't been anyone here for a long time.  Over in
    the corner there is a large open trophy case.
    There is a large rug covering most of the floor.
    There is a large gas lamp.

          What would you like to do?
          > GET LAMP
```

```
Copyright (c) 1981, 1982, 1983 Infocom, Inc. All rights reserved.
ZORK is a registered trademark of Infocom, Inc.
Revision 88 / Serial number 840726

West of House
You are standing in an open field west of a white house, with a boarded front
door.
There is a small mailbox here.

>open mailbox
Opening the small mailbox reveals a leaflet.

>take the leaflet
Taken.

>read leaflet

                                   low cunning. In it you will explore
                                  seen by mortals. No computer should be
```

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip 0, Program:   _ZORK1
Clearing                                      Score: 0         Moves: 4
There is a small mailbox here.

>go north
North of House
You are facing the north side of a white house. There is no door here, and all
the windows are boarded up. To the north a narrow path winds through the trees.

>go north
Forest Path
This is a path winding through a dimly lit forest. The path heads north-south
here. One particularly large tree with some low branches stands at the edge of
the path.
You hear in the distance the chirping of a song bird.

>go north
Clearing
You are in a clearing, with a forest surrounding you on all sides. A path
leads south.
On the ground is a pile of leaves.

>jump into leaves
Very good. Now you can go to the second grade.

>
```

Learning Outcomes:

Programming Fundamentals:
Students gain a deeper understanding of core programming concepts such as variables, data types, loops, conditionals, functions, and arrays. They learn how to apply these concepts effectively to implement game mechanics and logic.

Problem-Solving Skills:
Designing and implementing a text-based game requires critical thinking and problem-solving skills. Students learn how to break down complex problems into manageable tasks, show requirements, and devise solutions.

Algorithmic Thinking:
Developing a game involves designing algorithms to manage game mechanics, player interactions, and decision-making processes. Students practice algorithmic thinking by designing efficient and logical solutions to various game scenarios.

User Input Handling:
Managing user input is crucial in game development. Students learn how to manage user input securely, confirm input data, and provide meaningful feedback to players.

Game Design Principles:
While creating a text-based game, students explore fundamental game design principles such as player engagement, feedback loops, meaningful choices, and game balance. They gain insights into what makes a game enjoyable and engaging for players.

Debugging and Testing:
Debugging and testing are integral parts of software development. Students learn how to find and fix bugs, conduct unit testing, and ensure their game functions correctly across different scenarios.

Project Management:
Developing a game involves planning, organizing, and managing resources effectively. Students learn project management skills such as task prioritization, time management, version control, and collaboration if working in teams.

Creativity and Innovation:
Game development encourages creativity and innovation. Students have the freedom to come up with unique game ideas, design interesting gameplay mechanics, and implement creative solutions to enhance player experience.

Data Structures:
Students can learn about different data structures such as linked lists, queues, stacks, and trees by implementing them in the context of game development. For instance, they may use a stack to manage game states or a queue to oversee player actions.

User Interface Design:
While text-based games do not have graphical user interfaces (GUIs), students can still explore user interface design principles such as clarity, consistency, and usability in the context of text-based interactions. They learn how to present information effectively to players through textual cues and prompts.

Memory Management:
Game development often involves managing memory efficiently, especially for larger projects or when working with dynamic data structures. Students learn about memory allocation, deallocation, pointers, and memory optimization techniques in C++.

Error Handling and Exception Handling:
Students gain experience in handling errors, exceptions, and edge cases that may arise during game execution. They learn how to implement robust error handling mechanisms to prevent crashes and improve the overall stability of their game.

Game Balancing and Tuning:
Balancing gameplay elements such as difficulty levels, resource management, and progression curves is an essential aspect of game design. Students learn how to analyze and adjust game parameters to create a balanced and enjoyable gaming experience.

Documentation and Communication:
Effective communication and documentation are vital skills in software development. Students learn how to document their code, write clear comments, and communicate technical concepts to team members or potential users.

Player Psychology and Engagement:
Creating a game requires understanding player behavior, motivations, and engagement strategies. Students can explore principles of psychology and game theory to design compelling narratives, meaningful choices, and rewarding feedback systems in their games.

Cross-Platform Development:
While text-based games are typically platform-independent, students can still learn about cross-platform development considerations and techniques. They may explore how to compile and run their C++ games on different operating systems or environments.

Community and Collaboration:
Game development often involves collaboration and participation in online communities or forums. Students can engage with peers, share their projects, receive feedback, and learn from others' experiences, fostering a sense of community and collaboration in the development process.

About Text-Based Games:

Text-based games, also known as interactive fiction or text adventures, are games that rely primarily on text to communicate with players and present gameplay. These games can vary widely in complexity, genre, and platform, but they all share the common trait of using text as the primary means of interaction.

History of Text-Based Games

Text-based games have a rich history dating back to the early days of computing. One of the earliest and most famous examples is "Adventure" (also known as "Colossal Cave Adventure" or just "ADVENT"), created by Will Crowther in the 1970s. This game, which featured a fantasy setting and exploration-based gameplay, laid the groundwork for many future text adventures.

The genre gained further popularity with the release of games like "Zork" and the development of interactive fiction platforms like Infocom's Z-Machine, which allowed for more complex storytelling and gameplay mechanics.

Gameplay and Mechanics

Text-based games typically involve reading descriptions of locations, objects, and characters presented in text form and responding to prompts by typing commands. These commands can range from simple directions ("go north") to complex actions ("use key to unlock door").

The game's parser interprets the player's input and responds accordingly, advancing the story, triggering events, and allowing the player to progress through the game world. Some text-based games feature branching narratives and multiple endings, adding depth and replay value.

Modern Text-Based Games

While the popularity of text-based games waned with the rise of graphical gaming in the 1980s and beyond, the genre never disappeared entirely. In fact, there has been a resurgence of interest in text-based games in recent years, fuelled by indie developers and interactive fiction communities.

Modern text-based games often leverage digital platforms such as web browsers, mobile devices, and resolute interactive fiction interpreters. They cover a wide range of genres, from classic fantasy and sci-fi adventures to interactive novels, mystery games, and experimental storytelling experiences.

Key Elements of Text-Based Games

Narrative Focus: Text-based games prioritize storytelling and player choice, often offering richly detailed narratives and immersive worlds.

Player Agency: Players have a high degree of agency in text-based games, as they can influence the story and outcomes through their decisions and actions.

Puzzle Solving: Many text adventures include puzzles and challenges that players must solve using logic, observation, and creative thinking.

Exploration: Text-based games often encourage exploration of their virtual worlds, with players uncovering secrets, discovering new locations, and interacting with diverse characters.

Conclusion

Text-based games have a rich history and continue to evolve as a unique and engaging form of interactive entertainment. Their focus on narrative, player agency, and creative expression makes them a compelling option for both players and developers interested in exploring immersive storytelling experiences.

About C++ and The Compiler

C++ is a powerful and widely used programming language that is an extension of the C programming language. Here are some key points about C++:

Object-Oriented Programming (OOP):
C++ is primarily an object-oriented programming language, which means it supports concepts like classes, objects, inheritance, encapsulation, and polymorphism. These features allow for better organization and management of code, making it easier to develop and maintain large-scale projects.

High-level and Low-level Programming:
C++ is often considered a high-level language due to its rich set of features and abstractions that make programming easier. However, it also provides low-level access to system resources, such as memory management, making it suitable for system-level programming.

Compiled Language:
C++ is a compiled language, which means that code written in C++ needs to be compiled into machine code before it can be executed. This compilation process results in faster execution compared to interpreted languages.

Standard Template Library (STL):
C++ includes a powerful library called the Standard Template Library (STL), which provides a collection of classes and functions for various purposes such as containers (e.g., vectors, lists, maps), algorithms (e.g., sorting, searching), and iterators.

Platform Independence:
C++ code can be compiled and executed on various platforms, including Windows, macOS, Linux, and others, making it a cross-platform language.

Performance:
C++ is known for its high performance and efficiency, making it suitable for applications where speed and resource utilization are critical, such as real-time systems, games, and high-performance software.

Community and Support:
C++ has a large and active community of developers, along with extensive documentation and resources available online. This makes it easier for programmers to learn, troubleshoot, and collaborate on C++ projects.

Use Cases:
C++ is used in a wide range of applications, including system software (operating systems, device drivers), applications software (games, desktop applications), embedded systems, high-performance computing, and more.

Visual Studio Code (VS Code) is a popular integrated development environment (IDE) developed by Microsoft. While VS Code itself is not a compiler, we can configure it to work with various compilers and build systems to compile and run code written in different programming languages, including C++.

Here are a few key points regarding the use of compilers with VS Code:

Compiler Selection: VS Code allows you to choose which compiler to use for C++ development. Common choices include GCC (GNU Compiler Collection) on Linux, Clang on macOS, and MinGW or Visual C++ on Windows. You can install the necessary compiler tools separately and configure VS Code to use them.

C++ Extensions: VS Code has extensions specifically designed for C++ development, such as "C/C++" by Microsoft or "C/C++ IntelliSense, debugging, and code browsing" by Microsoft. These extensions provide features like code completion, syntax highlighting, debugging capabilities, and integration with build systems.

Build Systems: VS Code supports various build systems for C++ projects, such as CMake, Makefiles, and others. You can create build configurations and tasks within VS Code to compile and build your C++ code using the chosen compiler.

Debugging: VS Code offers robust debugging support for C++ programs. You can set breakpoints, inspect variables, step through code, and analyze program execution using the integrated debugger in VS Code.

Configuration Files: Use a compiler with VS Code for C++ development, you typically need to configure settings and provide paths to compiler tools, include directories, libraries, and build commands. These configurations are often specified in JSON files such as c_cpp_properties.json for compiler settings and tasks.json for build tasks.

Platform Support: VS Code is a cross-platform IDE, meaning it runs on Windows, macOS, and Linux. You can set up C++ development environments with compilers and build tools on any of these platforms and use VS Code for coding, building, and debugging C++ applications.

About Idea Generation:

Gather Resources:
Include resources like rocks, sticks, and plants that the player can collect to craft items like torches, ropes, and tools.

Crafting System:
Implement a crafting system where the player can combine collected resources to create useful items for escaping the cave.

Combat Mechanics:
Introduce monsters or creatures in the cave that the player must fight using crafted weapons or tools.

Traps and Puzzles:
Include traps and puzzles that the player must solve to progress further in the cave, such as hidden switches, locked doors, and collapsing tunnels.

Math Challenges:
Design various math challenges for the player to solve, such as addition, subtraction, multiplication, and division problems.

Checkpoints:
Place checkpoints at intervals throughout the parkour course where the player can restart from if they fall due to a wrong math answer.

Increasing Difficulty:
Make the math questions progressively more challenging as the player advances through the course.

Timer or Score System:
Implement a timer or scoring system to track the player's performance and challenge them to complete the course quickly with minimal mistakes.

Instructions and Directions:
Provide the player with initial instructions about cardinal directions (north, south, east, west) or landmarks and task them with navigating through a series of locations.

Wrong Direction Consequences:
If the player gives the wrong direction, simulate consequences such as getting lost, encountering obstacles, or receiving warnings about going off course.

Limited Wrong Guesses:
Set a limit on the number of incorrect directions the player can give before being removed from the navigator job, adding a sense of urgency and strategy.

Visual Map or Compass:
Include visual aids like maps or compasses to help players make informed decisions about directions.

Tactical Decisions:
Present the player with strategic decisions during the war, such as deploying troops, launching attacks, negotiating treaties, or fortifying defences.

Consequences of Actions:
Show the consequences of each decision made by the player, including casualties, diplomatic outcomes, and impact on the overall war effort.

Resource Management:
Manage resources like troops, ammunition, supplies, and morale, requiring the player to balance offense, Defense, and sustainability.

Multiple Endings:
Depending on the player's decisions throughout the game, provide different endings such as victory, defeat, stalemate, or diplomatic resolution.

Pattern Recognition:
Create various patterns using shapes, numbers, or symbols that the player must analyze to identify the next pattern in the sequence.

Hint System:
Include a hint system that provides clues or partial patterns to assist the player in solving more puzzling pattern challenges.

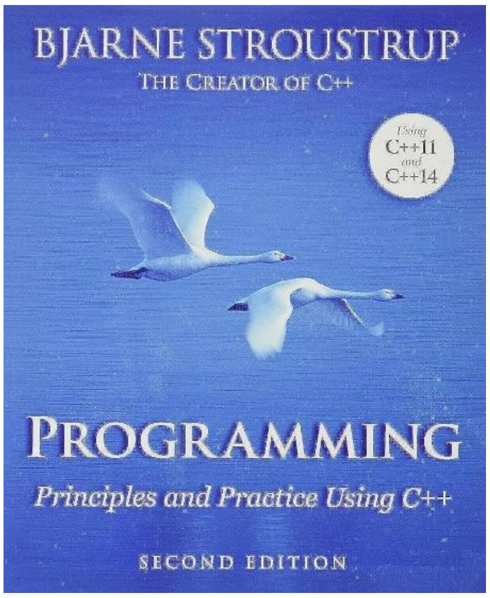
Progression and Difficulty:
Increase the complexity and diversity of patterns as the player advances through the game, evaluating their pattern recognition skills.

Score and Reward System:
Reward the player with points or rewards for each correct pattern guessed and track their progress toward the goal of guessing a certain number of patterns correctly within the given lives.

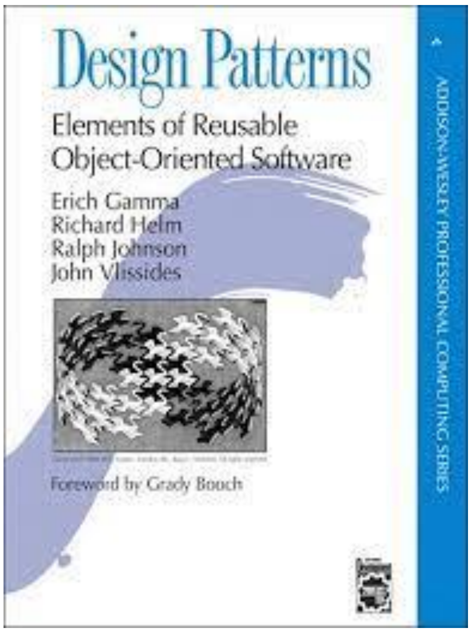
Reference:

Books:

1. "Programming: Principles and Practice Using C++" by Bjarne Stroustrup:
   This book by the creator of C++ provides a solid foundation in programming concepts using C++. It covers topics like variables, loops, functions, and classes, which are essential for game development.



2. "Beginning C++ Through Game Programming" by Michael Dawson:
   This book focuses on teaching C++ programming through the development of simple games. It covers basic game development concepts and techniques that can be applied to text-based games as well.


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

3. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides:
   This classic book introduces design patterns that are applicable to game development, including text-based games. Understanding design patterns can help you design robust and maintainable game code.

Online Tutorials and Courses:

1. Codecademy C++ Course:
   Codecademy offers an interactive C++ course that covers essential programming concepts. While it may not focus specifically on game development, it provides a solid introduction to C++ programming, which is crucial for creating games.

2. Udemy C++ Game Development Courses:
   Udemy has various courses specifically tailored to C++ game development. Look for courses that cover text-based games or basic game development concepts
   suitable for beginners.

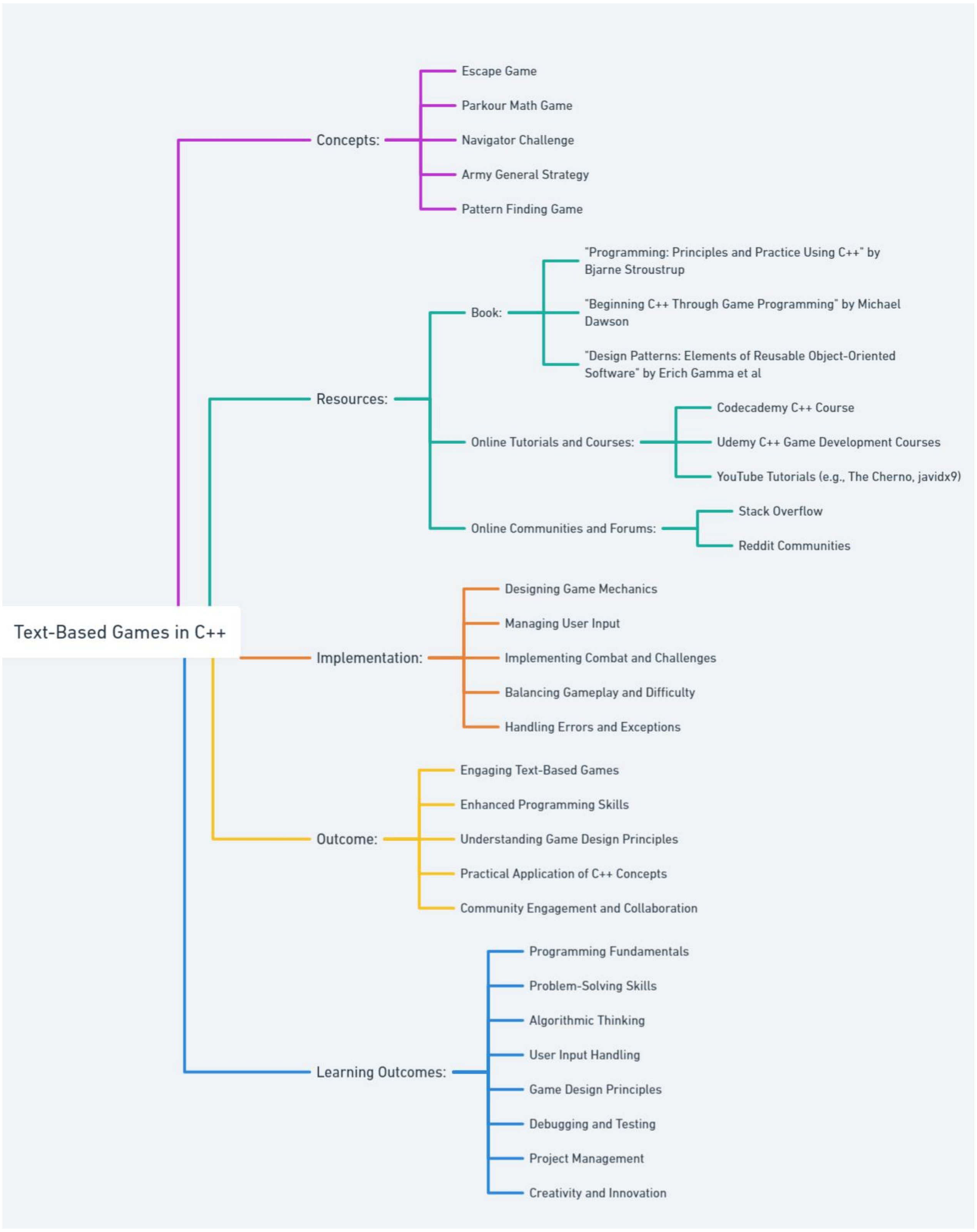
3. YouTube Tutorials:
   Many YouTube channels provide tutorials on C++ game development, including text-based games. Channels like The Cherno and javidx9 offer informative videos on game programming techniques and concepts.

Online Communities and Forums:

1. Stack Overflow:
   Stack Overflow is a valuable resource for asking programming-related questions and getting help from experienced developers. Search for existing threads or create new ones if you encounter specific issues or challenges in your game development journey.

2. Reddit Communities:
Subreddits like r/gamedev and r/learnprogramming have active communities of game developers and programmers who can provide guidance, feedback, and resources for text-based game development in C++.

Mind MAP



Text-Based Games in C++
Concepts:
Escape Game
Parkour Math Game
Navigator Challenge
Army General Strategy
Pattern Finding Game
Resources:
Book:
"Programming: Principles and Practice Using C++" by Bjarne Stroustrup
"Beginning C++ Through Game Programming" by Michael Dawson
"Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al
Online Tutorials and Courses:
Codecademy C++ Course
Udemy C++ Game Development Courses
YouTube Tutorials (e.g., The Cherno, javidx9)
Online Communities and Forums:
Stack Overflow
Reddit Communities
Implementation:
Designing Game Mechanics
Managing User Input
Implementing Combat and Challenges
Balancing Gameplay and Difficulty
Handling Errors and Exceptions
Outcome:
Engaging Text-Based Games
Enhanced Programming Skills
Understanding Game Design Principles
Practical Application of C++ Concepts
Community Engagement and Collaboration
Learning Outcomes:
Programming Fundamentals
Problem-Solving Skills
Algorithmic Thinking
User Input Handling
Game Design Principles
Debugging and Testing
Project Management
Creativity and Innovation

Research:

Game Development Concepts:

a. Text-Based Game Mechanics:
   1. Decision-Making: Research different decision-making structures such as branching narratives, multiple choices with consequences, and dynamic story progression based on player decisions.
   2. Puzzle-Solving: Explore puzzle design principles, including puzzle types (logical, spatial, riddles), difficulty progression, hints, and solutions.
   3. Resource Management: Study resource gathering, inventory systems, crafting mechanics, and economy balancing within text-based games.
   4. Combat Systems: Investigate combat mechanics such as turn-based battles, real-time combat, action sequences, and enemy AI interactions in text-based format.

b. Game Design Principles:
   1. Player Engagement: Understand techniques for engaging players through immersive storytelling, compelling characters, meaningful choices, and emotional impact.
   2. Feedback Loops: Learn about feedback mechanisms, including visual, auditory, and textual feedback to guide player actions and provide information.
   3. Game Balance: Explore methods for balancing difficulty, progression, rewards, challenges, and pacing to create a satisfying gameplay experience.
   4. User Experience (UX) Design: Research UX principles for text-based interfaces, including readability, clarity, navigation, input handling, and accessibility considerations.

c. Game Genres:
   1. Interactive Fiction: Study interactive fiction genres such as interactive novels, choose-your-own-adventure stories, text adventures, and interactive narratives.
   2. Adventure Games: Explore adventure game elements like exploration, puzzle-solving, story-driven gameplay, character interactions, and environmental storytelling.
   3. Role-Playing Games (RPGs): Research RPG mechanics like character customization, stat management, quest systems, progression systems, and narrative branching.

Programming Fundamentals:

a. C++ Basics:
   1. Variables and Data Types: Review different variable types (integers, floats, strings, Booleans) and data structures (arrays, vectors) used in C++ programming.
   2. Operators and Expressions: Understand arithmetic, relational, logical, and bitwise operators, along with expressions and operator precedence rules.
   3. Control Structures: Learn about loops (for, while, do-while), conditionals (if-else, switch-case), and control flow statements for program logic.
   4. Functions and Libraries: Explore function definitions, function prototypes, parameter passing, return values, and standard C++ libraries for common tasks (iostream, string, etc.).

b. Object-Oriented Programming (OOP):
   1. Classes and Objects: Understand class definitions, object instantiation, member variables, member functions (methods), and encapsulation principles.
   2. Inheritance and Polymorphism: Explore inheritance hierarchies, base classes, derived classes, method overriding, virtual functions, and polymorphic behaviour.
   3. Abstraction and Encapsulation: Study abstraction concepts for hiding implementation details, encapsulation for data protection, and access specifiers (public, private, protected).|

Game Design and Development Tools:

a. Game Engines and Libraries:
   1. SFML (Simple and Fast Multimedia Library): Learn SFML for graphics rendering, window management, input handling, audio playback, and basic game development functionalities.
   2. SDL (Simple DirectMedia Layer): Explore SDL for cross-platform development, multimedia handling, event-driven programming, and integrating graphics, audio, and input.
   3. Unity with C#: While Unity is primarily used for graphical games, explore Unity's text-based features for narrative games, text interactions, and UI design.

b. Integrated Development Environments (IDEs):

1. Visual Studio: Familiarize yourself with Visual Studio's features for C++ development, including code editor, debugger, project management, and IntelliSense for code completion.
2. Code::Blocks: Explore Code::Blocks for C++ programming, project organization, build configurations, and compiler integrations.
3. CLion: Learn CLion's capabilities for C++ development, code navigation, refactoring, version control integration, and productivity tools.

Educational Resources:

a. Books and Online Courses:
   1. C++ Programming Books: Choose books covering C++ basics, intermediate topics, advanced techniques, game development in C++, and specific areas like algorithms or data structures.
   2. Game Design and Development Courses: Enrol in online courses covering game design principles, game development workflows, C++ programming for games, and specific game genres.
   3. Tutorial Websites: Explore tutorial websites offering step-by-step guides, tutorials, exercises, and projects for learning C++, game development, and related topics.

b. Tutorials and Guides:
   1. Video Tutorials: Watch video tutorials on YouTube, educational platforms (like Udemy, Coursera), and game development channels for practical demonstrations, coding examples, and explanations.
   2. Documentation: Refer to official documentation for C++ programming, game engines, libraries, and development tools to understand APIs, usage instructions, and best practices.

Game Design Documentation:

a. Game Design Document (GDD):
   1. Content: Create a detailed GDD outlining game concept, story, characters, gameplay mechanics, UI/UX design, levels, progression, challenges, and game systems.
   2. Diagrams and Flowcharts: Use diagrams (flowcharts, UML diagrams) to visualize game mechanics, decision trees, state machines, level designs, and player interactions.

Community Engagement:

a. Game Development Communities:
   1. Forums and Discussion Boards: Join game development forums, communities (like Indie Game Devs, GameDev.net), and subreddits to ask questions, share insights, seek feedback, and connect with fellow developers.
   2. Social Media Groups: Follow game development groups, hashtags, and industry experts on social media platforms (Twitter, LinkedIn, Facebook) for updates, networking, and learning opportunities.
   3. Game Jams and Events: Participate in game jams, hackathons, conferences, workshops, and local meetups to collaborate, display projects, learn new skills, and get inspired by others' work.

Testing and Feedback:

a. Playtesting:
   1. Beta Testing: Conduct beta testing phases with a focus group or external testers to gather feedback on gameplay mechanics, user experience, bugs, balance issues, and overall player satisfaction.
   2. Iterative Development: Iterate on game design, mechanics, code optimizations, bug fixes, and feature enhancements based on playtest feedback, analytics, and usability testing results.

Legal and Copyright Considerations:

a. Intellectual Property (IP):
   1. Copyright Laws: Understand copyright laws, licenses, and permissions for game assets (artwork, music, sound effects, fonts), code libraries, third-party tools, and open-source components.
   2. Trademark and Branding: Consider trademarking game names, logos, and branding elements if needed, and ensure compliance with trademark regulations.
   3. Legal Agreements: Create legal agreements (End User License Agreements, Terms of Service, Privacy Policies) for players, outlining rights, responsibilities, data usage, and dispute resolutions.

Concept 1:

In this text-based escape game set in a deep cave, players take on the role of a stranded explorer who must navigate a labyrinthine underground world to find freedom. To survive, players gather resources such as rocks, sticks, and plants, strategically crafting items like torches for illumination, ropes for climbing, and tools for excavation. Along the way, they encounter menacing monsters and cunning traps that challenge their wit and courage. Players must use their crafted items wisely to outsmart foes, solve puzzles, and unlock hidden passages leading to eventual escape. The game immerses players in a thrilling adventure where resourcefulness, quick thinking, and strategic decision-making are key to overcoming obstacles and emerging victorious from the depths of the cave. As they progress, they uncover the dark secrets of the cave and face moral dilemmas that test their resolve and shape the outcome of their escape.

Concept 2:

In this unique text-based parkour game, players are not only challenged by physical obstacles but also by mathematical puzzles. As they navigate through a dynamic parkour course, players encounter various platforms and gaps, each requiring a specific jump distance to cross. To determine the correct jump distance, players are presented with math questions that correspond to the gap's width. Upon receiving a math question, players must solve it accurately to calculate the precise jump distance needed. Correct answers propel players to the next platform, where new challenges await. However, failure to solve the math question correctly results in a misjudged jump, leading to a perilous fall. Players then restart from the previous checkpoint, emphasizing the importance of both mathematical acumen and agility in parkour. The game's progressive difficulty keeps players engaged, with each level introducing more complex math problems and challenging parkour sequences. Additionally, strategic checkpoint placement ensures a fair balance between risk and reward, encouraging players to strategize their approach to overcome obstacles efficiently. Through this innovative fusion of math challenges and parkour gameplay, players sharpen their problem-solving skills while experiencing an exhilarating and educational adventure.

Concept 3:

The concept for this code encompasses a multi-faceted text-based adventure that unfolds in stages, starting from an interview scenario and progressing to real-world navigation challenges. Players engage in a journey of skill development, beginning with a test that assesses their ability to recognize patterns and make informed choices. Upon successfully passing the test, players transition into the role of a navigator tasked with guiding passengers through a series of complex routes. Each decision made by the player, from answering questions to choosing directions, carries weight and contributes to the overall narrative. The game's structure, combining cognitive challenges with strategic decision-making, creates an immersive and educational experience that encourages players to think critically, adapt to changing situations, and explore the consequences of their actions. As players navigate through various scenarios and levels, they not only sharpen their navigational skills but also experience the thrill and responsibility of being a competent navigator in a dynamic and evolving environment.

Concept 4:

In this high-stakes text-based strategy game set in a war between India and China, players step into the role of an army general tasked with making critical decisions that will shape the course of the conflict. Faced with dynamic challenges and rapid developments on the battlefield, players must think strategically and act decisively to outmanoeuvre the enemy and secure victory for their forces. Each decision carries weighty consequences, with wrong choices potentially leading to increased hostilities, casualties, or even the complete downfall of the army and the loss of the war. Conversely, making the right decisions, such as deploying forces strategically, conducting effective counterattacks, negotiating diplomatically, and managing resources efficiently, can turn the tide of battle and lead to a triumphant outcome. The game's immersive narrative, tactical depth, and risk-reward mechanics provide players with a thrilling and thought-provoking experience as they navigate the complexities of military leadership and strive to emerge as victorious generals in the face of adversity.

Concept 5:

In this engaging pattern finding game, players are challenged to decipher and predict patterns presented to them. With each level, players are given a specific pattern and tasked with identifying the next logical step in the sequence. The game tests players' pattern recognition skills, logical reasoning, and deductive abilities as they strive to crack each pattern successfully. To add excitement and challenge, players are allotted three lives throughout the game. Incorrect guesses deduct from these lives, and if a player makes three consecutive wrong guesses, they lose a life. The goal is to correctly guess at least eight out of the ten patterns presented to achieve victory. Players must rely on observation, analysis, and critical thinking to uncover the underlying logic behind each pattern and make informed guesses. The game's progression introduces increasingly complex patterns, requiring players to adapt and refine their pattern recognition strategies to succeed. With its blend of cognitive challenge, risk management, and goal-oriented gameplay, the pattern finding game offers players a stimulating and rewarding experience as they strive to master the art of pattern recognition and emerge victorious within the allotted lives and pattern sequences.

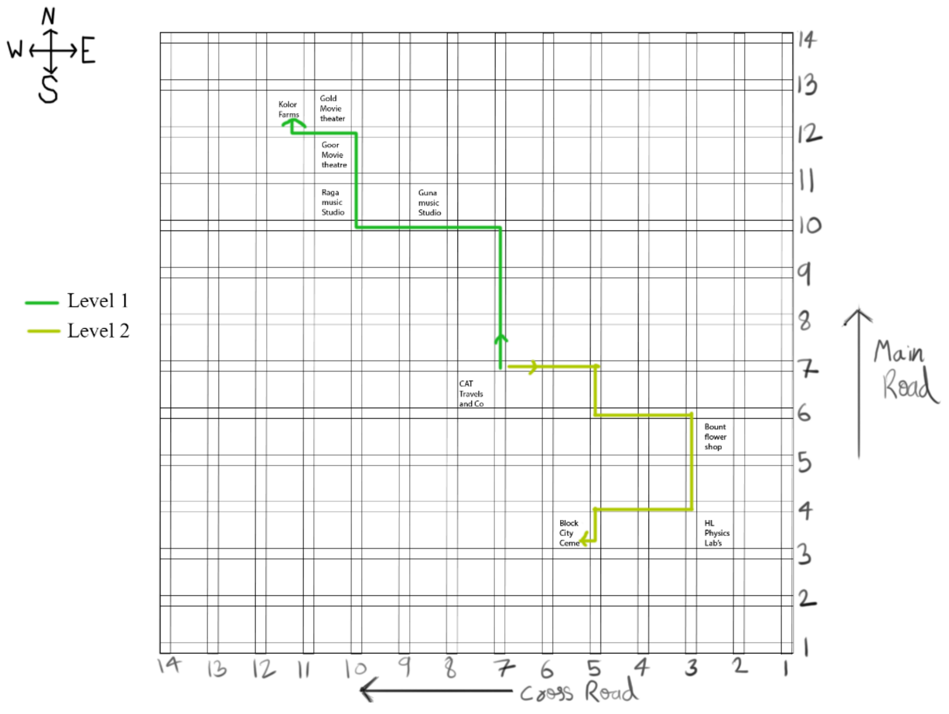
Final Concept ("Become a Navigator"):

This game I have chosen and made is a puzzle type text-based game where the player must take an interview in the starting of the game. To progress to the next level which is the first assignment the player must pass the interview. The game is set is a fictional city named Block City where the city is totally constructed in a proper blocks form. I have only done with the first level as the code lines went over 2000. The Second level was planed in the map but was not implemented.

The player if passed the interview, they will be given few instructions to follow these instructions are only shown ones, so the player must remember the instructions throughout the level.

The player first starts from the office and is assigned as a guide. The player navigates through the block city with the given instructions and wins the level. If the player takes a wrong turn, the player has an option to take a U-Turn but if the player takes 2 wrong directions, then the player is OUT and is removed from the job.

In this Text-Based game I have used a "type" function which outputs the text in a typewriter form, and also used some C++ built-in functions to change the text colour.



MAP for Block City

Code and Explanation:

```
#include <iostream>
#include <windows.h>
#include <conio.h>
#include <string>
```

These are the C++ Libraries used for building this Text-Based game.

```
using namespace std;
```

"using namespace std" means that we can use names for objects and variables from the standard library.

```
string Name;
char yorn,chooseabcd;
int marks = 0,Wrong_direction = 0;
char replayorexit;
string stri;
```

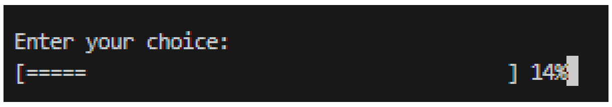
The above variables are used throughout the program to handle data and user input.
The Name string stores the name of the player, the yorn, chooseabcd and replayorexit char type is used for user input, the marks and Wrong_direction is used to store values that are later used to decide the game progression.

```
void typer(string c)
{
    int a=0;
    int b = c.size();
    while(b>=a)
    {
        cout << c[a];
        int st = rand() % 20 + 20;
        Sleep(st);
        a++;
    }
    cout<<endl;
}
```

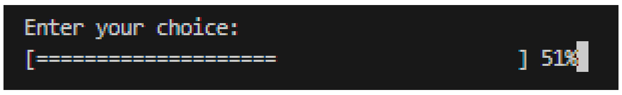
In this block of code I have defined a function named "typer", which helps the text output to print in typewriter style. This function takes in a string and outputs one letter each for a random tick speed of 20 milliseconds to 40 milliseconds from the given string.

```
void loadingBar() {
    const int totalWidth = 40;
    const int delayMillis = 100;
    const int numIterations = 100;

    for(int i = 0; i <= numIterations; ++i) {

        int progress = (i * 100) / numIterations;

        int numEquals = (i * totalWidth) / numIterations;

        cout << "\r[";
        for(int j = 0; j < numEquals; ++j) {
            cout << "=";
        }
        for(int j = numEquals; j < totalWidth; ++j) {
            cout << " ";
        }
        cout << "] " << progress << "%";

        Sleep(delayMillis-75);
    }
    Sleep(500);
    cout << endl;
}
```

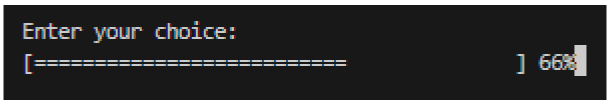
This code here is used to give a Game loading feel to the player.



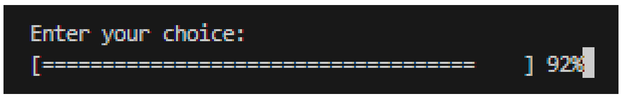
```
Enter your choice:
[=====                                   ] 14%
```



```
Enter your choice:
[====================                    ] 51%
```



```
Enter your choice:
[==========================              ] 66%
```



```
Enter your choice:
[====================================    ] 92%
```

```
 class Com_qus
{
    public:
    int Testguidstatwithcout()
    {
        cout << "We will be taking a test as part of the interview process please choose the correct answer for the following questions."<<endl<<"\nYou will be shown 5 words in a specific order for 3 seconds. Remember the order and choose the correct answer.\nA total of 3 questions will be given each question contains a maximum of 10 marks"<<endl<<"\nEnter only a , b , c , d or A , B , C , D"<<endl;
        return 0;
    }
    int Testguidstat()
    {
        typer("We will be taking a test as part of the interview process please choose the correct answer for the following questions.\nYou will be shown 5 words in a specific order for 3 seconds. Remember the order and choose the correct answer.\nA total of 3 questions will be given each question contains a maximum of 10 marks\nEnter only a , b , c , d or  A , B , C , D");
        return 0;
    }
    int Cross_road_qu(string cross_name)
    {
        typer("You have arrived at "+ cross_name +" Cross Road\nWhich way to 
go? \na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
        return 0;
    }
    int Movie_theatre_qu(string theatre_name)
    {
        typer("You have arrived at "+ theatre_name +" movie theatre\nWhich way 
to go? \na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
        return 0;
    }
    int Telephone_Booth_qu(string booth_name)
    {
        typer("You have arrived at "+ booth_name +" telephone booth\nWhich way 
to go? \na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
        return 0;
    }
    int Market_qu(string market_name)
    {
        typer("You have arrived at "+ market_name +" market\nWhich way to go? 
\na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
        return 0;
    }
    int Music_qu(string studio_name)
    {
        typer("You have arrived at "+ studio_name +" Music Studio\nWhich way 
to go? \na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
        return 0;
    }
    int others_qn(string place_full_name)
    {
        typer("You have arrived at "+ place_full_name +" \nWhich way to go? 
\na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
        return 0;
    }
};
```

The above code block is a Structure holding multiple function which is later called to reduce code repetition.

```
int main()
{
    Start:
    Wrong_direction = 0;
    system("cls");
    cout << "\033[33m\n\n\t\t\t\t\t\t\t\t\tBecome A 
Navigator\033[0m\n"<<endl<<"\n\033[32m\t\t\t\t\tStart(y/Y)\033[0m\t\t\t\t\t\t\
t\t\t\t\033[31mExit(n/N)\033[0m"<<endl<<"\n\nEnter your choice: ";
    yorn = _getch();
```

This is the main() function where the code starts running from. I have used the Built-In C++ function called “goto statement” to make the code more understandable and remove the use of loops like “while” and “for”.

```
cout << "\033[33m\n\n\t\t\t\t\t\t\t\t\tBecome A Navigator\033
```

In the above line I have used the escape sequences to control the formatting and colour of the text output in C++.

- "\033[33m" This is an ANSI escape code that changes the text colour to yellow (33m represents the colour code for yellow in ANSI escape sequences).
- "\033" This is the escape sequence to reset the text formatting and colour.

```
if(yorn == 'y' || yorn == 'Y')
    {
        cout << "\n\033[34mYou pressed \033[0m"<<yorn<< " ";
        Sleep(1000);
        loadingBar();
    }
    else if(yorn == 'n' || yorn == 'N')
    {
        cout << "\n\033[34mYou pressed \033[0m"<< yorn << " " << endl;
        Sleep(1000);
        goto Exit;
    }
    else
    {
        system("cls");
        cout << "\nWrong Input try again...";
        Sleep(1000);
        goto Start;
    }
    system("cls");
```

This above code block checks the user input in the Menu screen. I have also used "system("cls")" function to clear the whole console and give a smoother gameplay and "Sleep()" function is used to give delay(Like a Pause) in between the gameplay.

```
Com_qus ask;
    typer("\nNarrator: You are looking for a job in an travel agency in Block
City. And have got an interview in CAT Travels and Co.\nWelcome to CAT Travels
and Co.\nPlease Provide your name: ");
    cin >> Name;
    typer("\nPlease be seatted we will take your interview shortly");
    Sleep(3000);
    system("cls");
    typer("\nHello, " + Name );
    testcontinuegoto:
    ask.Testguidstat();
    typer("\nPress \'y/Y\' to continue and \'n/N\' to Quit");
    yorn = _getch();
    if(yorn == 'y' || yorn == 'Y')
    {
        system("cls");
        typer("\nStarting test is 3 secounds");
        Sleep(1000);
        system("cls");
        cout << "\nStarting test is 2 secounds";
        Sleep(1000);
        system("cls");
        cout << "\nStarting test is 1 secounds";
        Sleep(1000);
        system("cls");

    }
    else if(yorn == 'n' || yorn == 'N')
    {
        goto Exit;
    }
    else
    {
        typer("\nWrong Input... Try again...");
        Sleep(1000);
        system("cls");
        goto testcontinuegoto;
    }
    errorstarttestagain:
    typer("\nGet ready...");
    Sleep(3000);
```

Here I have initialized the ask variable to the "Com_qus()" structure which is used later to call the function in it (Like the "Testguidstat" function).

```
typer("\nQuestion 1: Remember the order of the following words:
\n\n\033[33mDistance - Reached - Journy - Travel - Packing\033[0m");
    Sleep(3000);
    TestoneWronginput:
    system("cls");
    ask.Testguidstatwithcout();
    cout << "\nA) Distance - Journey - Packing - Travel - Reached"<<endl;
    cout << "B) Packing - Journey - Distance - Travel - Reached"<<endl;
    cout << "C) Distance - Reached - Journey - Travel - Packing"<<endl;
    cout << "D) Distance - Travel - Journey - Packing - Reached"<<endl;
    typer("Enter your answer: ");
    chooseabcd = _getch();
    if (chooseabcd=='a' || chooseabcd=='A')
    {
        marks += 4;
    }
    else if (chooseabcd=='b' || chooseabcd=='B')
    {
        marks +=2;
    }
    else if (chooseabcd=='c' || chooseabcd=='C')
    {
        marks +=10;
    }
    else if (chooseabcd=='d' || chooseabcd=='D')
    {
        marks +=4;
    }
    else
    {
        goto TestoneWronginput;
    }
    system("cls");
```

This Code Block is the interview prosses the player is asked 3 questions in total. The interview is that the player is given 5 different words and the player is given 3 seconds to remember the word order, after that the player is give 4 options to choose from and the marks are given accordingly

```
if(marks >= 20)
    {
        typer("\nCongratulations...You have passed the interview with a score
of "+ stri +" out of 30\nWelcome to CAT Travels and Co. \nFeel free to start
your job from today");
        Sleep(10000);
    }
    else if(marks < 20)
    {
        typer("\nSorry for this news but you failed the test with the score of
"+ stri +"out of 30 \nBetter luck next time");
        Sleep(3000);
        system("cls");
        goto End;
    }
    else
    {
        typer("\nSomething is wrong...Counting error accured \nPLEASE try the
test again");
        Sleep(3000);
        goto errorstarttestagain;
    }
```

This is how the marks are seen and the player is selected to the job. If the player fails the interview the player is out.

```
typer("\nNarrator: You start your job now by taking your first
assignment\nHello, "+Name+"\nYou will be given a few instruction and be sure
to remember it \nYou will require this instruction to reach you
destination.\nYou will be takeing few pasengers on this specific rout as they
want to explore this places\nRemember you only get 2 chances which means when
given 2 wrong directions you will be removed from the job\nPress \'y/Y\' to
continue and \'n/N\' to Quit");
    yorn = _getch();
    if(yorn == 'y' || yorn == 'Y')
    {
        typer("\nLet's start you first assinment");
        Sleep(2000);
    }
    else if(yorn == 'n' || yorn == 'N')
    {
        goto Exit;
    }
    else
    {
        typer("\nWrong Input... Try again...");
        Sleep(1000);
        system("cls");
        goto tryagainfirstjob;
    }
```

Here the player is asked if they want to continue playing the game and start there job as a Navigator.

```
typer("\nFirst head North from 7th main 7th cross road\nThen turn to the 3rd
left\nWhen you arrive at a Music Store named Raga Music studio Turn
right\nThen you will arrive at Gold Movie Theatre, turn left there.\nYou will
arrive at your destination.\nPress \'y/Y\' to continue and \'n/N\' to Quit");
    yorn = _getch();
    if(yorn == 'y' || yorn == 'Y')
    {
        typer("\nLet's start your at 7th main 7th cross road");
        Sleep(2000);
    }
    else if(yorn == 'n' || yorn == 'N')
    {
        goto Exit;
    }
    else
    {
        typer("\nWrong Input... Try again...");
        Sleep(1000);
        system("cls");
        goto FirstInstruction;
    }
```

This Code Block is welcoming the player to the job and giving them there first assignment and is give time to learn the route to travel. The player can take how much time they want here to understand the routs.

```
typer("\nQuestion: Which way to go? ");
    typer("\nA) North");
    typer("B) East");
    typer("C) West");
    typer("D) South");
    typer("Enter your answer: ");
```

Start of level 1 the player is asked which direction to start from

```
if(chooseabcd == 'a' || chooseabcd == 'A')
    {
        typer("Correct Direction");
        Sleep(3000);
        eightmainseventhcross:
        system("cls");
        ask.Cross_road_qu("8th main 7th");
        typer("Enter your answer: ");
        chooseabcd = _getch();
        cout << chooseabcd << endl;
else if(chooseabcd == 'b' || chooseabcd == 'B')
    {
        typer("Wrong Direction");
        Sleep(3000);
        Wrong_direction++;
        if(Wrong_direction>=2)
        {
            goto End;
        }
        Eastturngoto:
        Sleep(3000);
        system("cls");
        ask.Cross_road_qu("7th Main 6th");
        typer("Enter your answer");
        chooseabcd = _getch();
        cout << chooseabcd << endl;
else if(chooseabcd == 'c' || chooseabcd == 'C')
    {
        typer("Wrong Direction");
        Sleep(3000);
        Wrong_direction++;
        if(Wrong_direction>=2)
        {
            goto End;
        }
        Westturngoto:
        Sleep(3000);
        system("cls");
        ask.Cross_road_qu("7th Main 8th");
        typer("Enter your answer");
        chooseabcd = _getch();
        cout << chooseabcd << endl;
else if(chooseabcd == 'd' || chooseabcd == 'D')
    {
        typer("Wrong Direction");
        Sleep(3000);
        Wrong_direction++;
        if(Wrong_direction>=2)
        {
            goto End;
        }
        Southturngoto:
        Sleep(3000);
        system("cls");
        ask.Cross_road_qu("6th Main 7th");
        typer("Enter your answer");
        chooseabcd = _getch();
        cout << chooseabcd << endl;
else
        {
            typer("\nWrong Input... Try again...");
            Sleep(1000);
            system("cls");
            goto Officeloc;
        }
```

This code checks if the direction given is write or wrong, if it a wrong direction each if and else if code has another if and else if and those have another if and else if statements in them as we are giving the player the option to correct or take two wrong directions and get out. In ever if and else if code block of wrong direction the wrong direction is checked and returns to End goto statement if the player is given 2 wrong directions

```
if(chooseabcd == 'a' || chooseabcd == 'A')
{
    typer("Correct Direction");
    Sleep(3000);
    system("cls");
    goldmovietheatre:
    ask.Movie_theatre_qu("Gold");
    typer("Enter your answer: ");
    chooseabcd = _getch();
    cout << chooseabcd << endl;
    if(chooseabcd == 'b' || chooseabcd == 'B')
    {
        typer("Correct Direction");
        typer("You have reached your destination");
        Sleep(3000);
        system("cls");
    }
    else if(chooseabcd == 'a' || chooseabcd == 'A')
    {
        typer("Wrong Direction");
        Sleep(3000);
        Wrong_direction++;
        if(Wrong_direction>=2)
        {
            goto End;
        }
        thirteenthmaintenthcross:
        system("cls");
        ask.Cross_road_qu("13th Main 10th");
        typer("Enter your answer: ");
        chooseabcd = _getch();
        cout << chooseabcd << endl;
        if(chooseabcd == 'a'||chooseabcd == 'A')
        {
            typer("Wrong Direction");
            Sleep(3000);
            Wrong_direction++;
            system("cls");
            if(Wrong_direction>=2)
            {
                goto End;
            }
        }
        else if(chooseabcd == 'b'||chooseabcd == 'B')
        {
            typer("Wrong Direction");
            Sleep(3000);
            Wrong_direction++;
            system("cls");
            if(Wrong_direction>=2)
            {
                goto End;
            }
        }
        else if(chooseabcd == 'c'||chooseabcd == 'C')
        {
            typer("Wrong Direction");
            Sleep(3000);
            Wrong_direction++;
            system("cls");
            if(Wrong_direction>=2)
            {
                goto End;
            }
        }
        else if(chooseabcd == 'd'||chooseabcd == 'D')
        {
            typer("Correct Direction");
            typer("Taking U-Turn");
            Sleep(3000);
            system("cls");
            goto goldmovietheatre;
        }
        else
        {
            typer("\nWrong Input... Try again...");
            Sleep(1000);
            system("cls");
            goto thirteenthmaintenthcross;
        }
    }
    else if(chooseabcd == 'c' || chooseabcd == 'C')
        {
            typer("Wrong Direction");
            Sleep(3000);
            Wrong_direction++;
            if(Wrong_direction>=2)
            {
                goto End;
            }
            twelvethmainninethcross:
            system("cls");
            ask.Cross_road_qu("12th Main 9th");
            typer("Enter your answer: ");
            chooseabcd = _getch();
            cout << chooseabcd << endl;
            if(chooseabcd == 'a'||chooseabcd == 'A')
            {
                typer("Wrong Direction");
                Sleep(3000);
                Wrong_direction++;
                system("cls");
                if(Wrong_direction>=2)
                {
                    goto End;
                }
            }
            else if(chooseabcd == 'b'||chooseabcd == 'B')
            {
                typer("Wrong Direction");
                Sleep(3000);
                Wrong_direction++;
                system("cls");
                if(Wrong_direction>=2)
                {
                    goto End;
                }
            }
            else if(chooseabcd == 'c'||chooseabcd == 'C')
            {
                typer("Wrong Direction");
                Sleep(3000);
                Wrong_direction++;
                system("cls");
                if(Wrong_direction>=2)
                {
                    goto End;
                }
            }
            else if(chooseabcd == 'd'||chooseabcd == 'D')
            {
                typer("Correct Direction");
                typer("Taking U-Turn");
                Sleep(3000);
                system("cls");
                goto goldmovietheatre;
            }
            else
            {
                typer("\nWrong Input... Try again...");
                Sleep(1000);
                system("cls");
                goto tenthmainsixthcross;
            }
        }
    else if(chooseabcd == 'd' || chooseabcd == 'D')
    {
        typer("Wrong Direction");
        Sleep(3000);
        Wrong_direction++;
        if(Wrong_direction>=2)
        {
            goto End;
        }
        system("cls");
        goto goormovietheatre;
    }
    else
    {
        typer("\nWrong Input... Try again...");
        Sleep(1000);
        system("cls");
        goto goldmovietheatre;
    }
}
```

There are above 200 if and else if statement and up to 20 if and else if statement for the second nested level of the if and else if statements.

```
End:
    if(Wrong_direction>=2)
    {
        system("cls");
        typer("You have given 2 wrong directions" );
        typer("You are removed from the job!!" );
        Sleep(3000);
    }
```

Whenever the “End” goto function is called the wrong direction is checked again the ensure the number of wrong directions given is 2 or greater.

```
typer("GAME OVER\n\nDo you want to Replay the game or EXIT");
    typer("Press \nR for REPLAY\nE for EXIT");
    replayorexit = _getch();
    replayorexitgoto:
    if(replayorexit == 'r'||replayorexit == 'R')
    {
        goto Start;
    }
    else if(replayorexit == 'e'||replayorexit == 'E')
    {
        Exit:
        system("cls");
        typer("\nExiting in 5 secounds");
        Sleep(1000);
        system("cls");
        cout << "\nExiting in 4 secounds";
        Sleep(1000);
        system("cls");
        cout << "\nExiting in 3 secounds";
        Sleep(1000);
        system("cls");
        cout << "\nExiting in 2 secounds";
        Sleep(1000);
        system("cls");
        cout << "\nExiting in 1 secounds";
        Sleep(1000);
        system("cls");
        return 0;
    }
    else
        {
            typer("\nWrong Input... Try again...");
            Sleep(1000);
            system("cls");
            goto replayorexitgoto;
        }
}
```

Hear the player is shown the game over screen and asked if they want to retry the game or exit.

About the goto Statement:

The goto statement in C++ is a control flow statement that allows you to transfer the program's control to a labelled statement within the same function or code block. It's generally considered a low-level control flow mechanism and is often discouraged in modern programming practices due to its potential for creating unreadable and difficult-to-maintain code. However, understanding how goto works can be helpful for understanding older code or certain specific situations.

```
label:
goto label;
```

Where label is an identifier followed by a colon (:) that marks the location where the control flow will jump to when the goto statement is encountered.

Full Code:

```
#include <iostream>
#include <windows.h>
#include <conio.h>
#include <string>
using namespace std;
string Name;
char yorn,chooseabcd;
int marks = 0,Wrong_direction = 0;
char replayorexit;
string stri;
void typer(string c)
{
int a=0;
int b = c.size();
while(b>=a)
{
cout << c[a];
int st = rand() % 20 + 20;
Sleep(st);
a++;
}
cout<<endl;
}
void loadingBar() {
const int totalWidth = 40;
const int delayMillis = 100;
const int numIterations = 100;

for(int i = 0; i <= numIterations; ++i) {

int progress = (i * 100) / numIterations;

int numEquals = (i * totalWidth) / numIterations;

cout << "\r[";
for(int j = 0; j < numEquals; ++j) {
cout << "=";
}
for(int j = numEquals; j < totalWidth; ++j) {
cout << " ";
}
cout << "] " << progress << "%";

Sleep(delayMillis-75);
}
Sleep(500);
cout << endl;
}
class Com_qus
{
public:
int Testguidstatwithcout()
{
```

```
cout << "We will be taking a test as part of the interview process please
choose the correct answer for the following questions."<<endl<<"\nYou will be
shown 5 words in a specific order for 3 seconds. Remember the order and choose
the correct answer.\nA total of 3 questions will be given each question
contains a maximum of 10 marks"<<endl<<"\nEnter only a , b , c , d or  A , B ,
C , D"<<endl;
return 0;
}
int Testguidstat()
{
typer("We will be taking a test as part of the interview process please choose
the correct answer for the following questions.\nYou will be shown 5 words in
a specific order for 3 seconds. Remember the order and choose the correct
answer.\nA total of 3 questions will be given each question contains a maximum
of 10 marks\nEnter only a , b , c , d or  A , B , C , D");
return 0;
}
int Cross_road_qu(string cross_name)
{
typer("You have arrived at "+ cross_name +" Cross Road\nWhich way to go? \na)
Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
return 0;
}
int Movie_theatre_qu(string theatre_name)
{
typer("You have arrived at "+ theatre_name +" movie theatre\nWhich way to go?
\na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
return 0;
}
int Telephone_Booth_qu(string booth_name)
{
typer("You have arrived at "+ booth_name +" telephone booth\nWhich way to go?
\na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
return 0;
}
int Market_qu(string market_name)
{
typer("You have arrived at "+ market_name +" market\nWhich way to go? \na) Go
straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
return 0;
}
int Music_qu(string studio_name)
{
typer("You have arrived at "+ studio_name +" Music Studio\nWhich way to go?
\na) Go straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
return 0;
}
int others_qn(string place_full_name)
{
typer("You have arrived at "+ place_full_name +" \nWhich way to go? \na) Go
straight\nb) Turn left\nc) Turn right\nd) Take a U-Turn\n");
return 0;
}
};
int main()
{
```

```
Start:
Wrong_direction = 0;
system("cls");
cout << "\033[33m\n\n\t\t\t\t\t\t\t\t\tBecome A
Navigator\033[0m\n"<<endl<<"\n\033[32m\t\t\t\t\tStart(y/Y)\033[0m\t\t\t\t\t\t\
t\t\t\t\033[31mExit(n/N)\033[0m"<<endl<<"\n\nEnter your choice: ";
yorn = _getch();
if(yorn == 'y' || yorn == 'Y')
{
cout << "\n\033[34mYou pressed \033[0m"<<yorn<< " ";
Sleep(1000);
loadingBar();
}
else if(yorn == 'n' || yorn == 'N')
{
cout << "\n\033[34mYou pressed \033[0m"<< yorn << " " << endl;
Sleep(1000);
goto Exit;
}
else
{
system("cls");
cout << "\nWrong Input try again...";
Sleep(1000);
goto Start;
}
system("cls");
Com_qus ask;
typer("\nNarrator: You are looking for a job in an travel agency in Block
City. And have got an interview in CAT Travels and Co.\nWelcome to CAT Travels
and Co.\nPlease Provide your name: ");
cin >> Name;
typer("\nPlease be seatted we will take your interview shortly");
Sleep(3000);
system("cls");
typer("\nHello, " + Name );
testcontinuegoto:
ask.Testguidstat();
typer("\nPress \'y/Y\' to continue and \'n/N\' to Quit");
yorn = _getch();
if(yorn == 'y' || yorn == 'Y')
{
system("cls");
typer("\nStarting test is 3 secounds");
Sleep(1000);
system("cls");
cout << "\nStarting test is 2 secounds";
Sleep(1000);
system("cls");
cout << "\nStarting test is 1 secounds";
Sleep(1000);
system("cls");

}
else if(yorn == 'n' || yorn == 'N')
{
```

```
goto Exit;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto testcontinuegoto;
}
errorstarttestagain:
typer("\nGet ready...");
Sleep(3000);
skip:
system("cls");
ask.Testguidstatwithcout();
typer("\nQuestion 1: Remember the order of the following words:
\n\n\033[33mDistance - Reached - Journy - Travel - Packing\033[0m");
Sleep(3000);
TestoneWronginput:
system("cls");
ask.Testguidstatwithcout();
cout << "\nA) Distance - Journey - Packing - Travel - Reached"<<endl;
cout << "B) Packing - Journey - Distance - Travel - Reached"<<endl;
cout << "C) Distance - Reached - Journey - Travel - Packing"<<endl;
cout << "D) Distance - Travel - Journey - Packing - Reached"<<endl;
typer("Enter your answer: ");
chooseabcd = _getch();
if (chooseabcd=='a' || chooseabcd=='A')
{
marks += 4;
}
else if (chooseabcd=='b' || chooseabcd=='B')
{
marks +=2;
}
else if (chooseabcd=='c' || chooseabcd=='C')
{
marks +=10;
}
else if (chooseabcd=='d' || chooseabcd=='D')
{
marks +=4;
}
else
{
goto TestoneWronginput;
}
system("cls");
ask.Testguidstatwithcout();
typer("\nQuestion 2: Remember the order of the following words:
\n\n\033[33mMoon - Sky - Stars - Night - Light\033[0m");
Sleep(3000);
TesttwoWronginput:
system("cls");
ask.Testguidstatwithcout();
cout << "\nA) Light - Moon - Stars - Night - Sky"<<endl;
cout << "B) Moon - Sky - Stars - Night - Light"<<endl;
cout << "C) Sky - Stars - Moon - Night - Light"<<endl;
cout << "D) Moon - Sky - Light - Night - Stars"<<endl;
typer("Enter your answer: ");
chooseabcd = _getch();
if (chooseabcd=='a' || chooseabcd=='A')
{
marks += 2;
}
else if (chooseabcd=='b' || chooseabcd=='B')
{
marks +=10;
}
else if (chooseabcd=='c' || chooseabcd=='C')
{
marks +=4;
}
else if (chooseabcd=='d' || chooseabcd=='D')
{
marks +=6;
}
else
{
goto TesttwoWronginput;
}
system("cls");
ask.Testguidstatwithcout();
typer("\nQuestion 3: Remember the order of the following
words:\n\n\033[33mBook - Pencil - Eraser - Paper - Pen\033[0m");
Sleep(3000);
TestthreeWronginput:
system("cls");
ask.Testguidstatwithcout();
cout << "\nA) Pen - Paper - Eraser - Pencil - Book"<<endl;
cout << "B) Eraser - Pen - Book - Paper - Pencil"<<endl;
cout << "C) Pencil - Eraser - Paper - Pen - Book"<<endl;
cout << "D) Book - Pencil - Eraser - Paper - Pen"<<endl;
typer("Enter your answer: ");
chooseabcd = _getch();
if (chooseabcd=='a' || chooseabcd=='A')
{
marks += 2;
}
else if (chooseabcd=='b' || chooseabcd=='B')
{
marks +=2;
}
else if (chooseabcd=='c' || chooseabcd=='C')
{
marks +=0;
}
else if (chooseabcd=='d' || chooseabcd=='D')
{
marks +=10;
}
else
```

```
{
goto TestthreeWronginput;
}
system("cls");
typer("\nTest Ended...");
Sleep(3000);
stri = to_string(marks);
if(marks >= 20)
{
typer("\nCongratulations...You have passed the interview with a score of "+
stri +" out of 30\nWelcome to CAT Travels and Co. \nFeel free to start your
job from today");
Sleep(10000);
}
else if(marks < 20)
{
typer("\nSorry for this news but you failed the test with the score of "+ stri
+"out of 30 \nBetter luck next time");
Sleep(3000);
system("cls");
goto End;
}
else
{
typer("\nSomething is wrong...Counting error accured \nPLEASE try the test
again");
Sleep(3000);
goto errorstarttestagain;
}
tryagainfirstjob:
system("cls");
typer("\nNarrator: You start your job now by taking your first
assignment\nHello, "+Name+"\nYou will be given a few instruction and be sure
to remember it \nYou will require this instruction to reach you
destination.\nYou will be takeing few pasengers on this specific rout as they
want to explore this places\nRemember you only get 2 chances which means when
given 2 wrong directions you will be removed from the job\nPress \'y/Y\' to
continue and \'n/N\' to Quit");
yorn = _getch();
if(yorn == 'y' || yorn == 'Y')
{
typer("\nLet's start you first assinment");
Sleep(2000);
}
else if(yorn == 'n' || yorn == 'N')
{
goto Exit;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tryagainfirstjob;
}
skiptol1:
```

```
system("cls");
FirstInstruction:
typer("\nFirst head North from 7th main 7th cross road\nThen turn to the 3rd
left\nWhen you arrive at a Music Store named Raga Music studio Turn
right\nThen you will arrive at Gold Movie Theatre, turn left there.\nYou will
arrive at your destination.\nPress \'y/Y\' to continue and \'n/N\' to Quit");
yorn = _getch();
if(yorn == 'y' || yorn == 'Y')
{
typer("\nLet's start your at 7th main 7th cross road");
Sleep(2000);
}
else if(yorn == 'n' || yorn == 'N')
{
goto Exit;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto FirstInstruction;
}
system("cls");
Officeloc:
typer("\nQuestion: Which way to go? ");
typer("\nA) North");
typer("B) East");
typer("C) West");
typer("D) South");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
Sleep(3000);
eightmainseventhcross:
system("cls");
ask.Cross_road_qu("8th main 7th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if (chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
Sleep(3000);
ninthmainseventhcross:
system("cls");
ask.Cross_road_qu("9th main 7th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if (chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
```

```cpp
Sleep(3000);
tenthmainseventhcross:
system("cls");
ask.Cross_road_qu("10th main 7th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if (chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Correct Direction");
Sleep(3000);
tenthmaineightcross:
system("cls");
ask.Cross_road_qu("10th main 8th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if (chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
Sleep(3000);
gunamusicstudio:
system("cls");
ask.Music_qu("Guna");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if (chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
Sleep(3000);
tenthmainninethcross:
system("cls");
ask.Cross_road_qu("10th main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
Sleep(3000);
ragamusicstudio:
system("cls");
ask.Music_qu("Raga");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Correct Direction");
Sleep(3000);
goormovietheatre:
system("cls");
ask.Movie_theatre_qu("Goor");
typer("Enter your answer: ");
chooseabcd = _getch();
```

```
cout << chooseabcd << endl;
if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Correct Direction");
Sleep(3000);
system("cls");
goldmovietheatre:
ask.Movie_theatre_qu("Gold");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Correct Direction");
typer("You have reached your destination");
Sleep(3000);
system("cls");
}
else if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
thirteenthmaintenthcross:
system("cls");
ask.Cross_road_qu("13th Main 10th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
```

```
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto goldmovietheatre;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto thirteenthmaintenthcross;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
twelvethmainninethcross:
system("cls");
ask.Cross_road_qu("12th Main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto goldmovietheatre;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tenthmainsixthcross;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto goormovietheatre;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto goldmovietheatre;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
```

```
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
eleventhmaineleventhcross:
system("cls");
ask.Cross_road_qu("11th Main 11th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
```

```
goto goormovietheatre;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto eleventhmaineleventhcross;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
eleventhmainninethcross:
system("cls");
ask.Cross_road_qu("11th Main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto goormovietheatre;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto eleventhmainninethcross;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto ragamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto goormovietheatre;
}
}
else if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
tenthmaineleventhcross:
system("cls");
ask.Cross_road_qu("10th Main 11th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
```

```
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto ragamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tenthmaineleventhcross;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
```

```
{
goto End;
}
ninthmaintenthcross:
system("cls");
ask.Cross_road_qu("9th Main 10th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto ragamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninthmaintenthcross;
```

```
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto tenthmainninethcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ragamusicstudio;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
ninethmainninethcross:
system("cls");
ask.Cross_road_qu("9th Main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
```

```
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto gunamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninethmainninethcross;
}
}
else if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
elevenmainninethcrossfromtentheight:
system("cls");
ask.Cross_road_qu("11th Main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
```

```
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto gunamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto elevenmainninethcrossfromtentheight;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto gunamusicstudio;
}
else
{
```

```
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tenthmainninethcross;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
ninethmaineightcross:
system("cls");
ask.Cross_road_qu("9th Main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
```

```
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto gunamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninethmaineightcross;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
elevenmainninethcross:
system("cls");
ask.Cross_road_qu("11th Main 9th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
```

```
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto gunamusicstudio;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto elevenmainninethcross;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto tenthmaineightcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto gunamusicstudio;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
ninthmaineightcrossfromteneight:
```

```
system("cls");
ask.Cross_road_qu("9th Main 8th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto tenthmaineightcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninthmaineightcrossfromteneight;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
```

```
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
elevenmaineightcross:
system("cls");
ask.Cross_road_qu("11th Main 8th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto tenthmaineightcross;
}
else
{
```

```
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto elevenmaineightcross;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto tenthmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tenthmaineightcross;
}
}
else if(chooseabcd == 'a' || chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
eleventhmainseventhcross:
system("cls");
ask.Cross_road_qu("11th Main 7th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
```

```
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto tenthmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto eleventhmainseventhcross;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
tenthmainsixthcross:
system("cls");
ask.Cross_road_qu("10th Main 6th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
```

```
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto tenthmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tenthmainsixthcross;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto ninthmainseventhcross;
}
else
{
```

```
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto tenthmainseventhcross;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
ninthmaineightcross:
system("cls");
ask.Cross_road_qu("9th Main 8th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
```

```
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto ninthmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninthmaineightcross;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
ninthmainsixthcross:
system("cls");
ask.Cross_road_qu("9th Main 6th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
```

```
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto ninthmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninthmainsixthcross;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto eightmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto ninthmainseventhcross;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
eightmaineightcross:
system("cls");
ask.Cross_road_qu("8th Main 8th");
typer("Enter your answer: ");
chooseabcd = _getch();
```

```
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto eightmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto eightmaineightcross;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Wrong_direction++;
if(Wrong_direction>=2)
{
```

```
goto End;
}
eightmainsixthcross:
system("cls");
ask.Cross_road_qu("8th Main 6th");
typer("Enter your answer: ");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn");
Sleep(3000);
system("cls");
goto eightmainseventhcross;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto eightmainsixthcross;
}
```

```
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
system("cls");
goto Officeloc;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto eightmainseventhcross;
}
}
else if(chooseabcd == 'b' || chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
Eastturngoto:
Sleep(3000);
system("cls");
ask.Cross_road_qu("7th Main 6th");
typer("Enter your answer");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
```

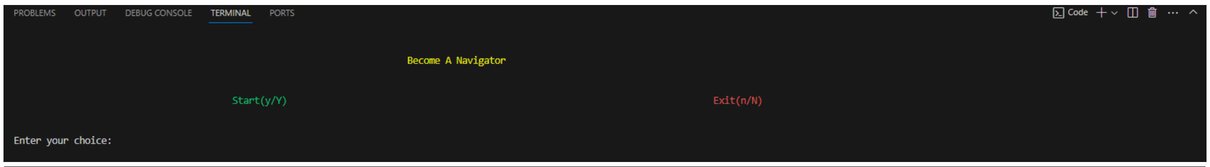
```
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn your Back at where you strated");
Sleep(3000);
system("cls");
goto Officeloc;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto Eastturngoto;
}
}
else if(chooseabcd == 'c' || chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
Westturngoto:
Sleep(3000);
system("cls");
ask.Cross_road_qu("7th Main 8th");
typer("Enter your answer");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
```

```
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn your Back at where you strated");
Sleep(3000);
system("cls");
goto Officeloc;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto Westturngoto;
}
}
else if(chooseabcd == 'd' || chooseabcd == 'D')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
if(Wrong_direction>=2)
{
goto End;
}
Southturngoto:
Sleep(3000);
system("cls");
ask.Cross_road_qu("6th Main 7th");
typer("Enter your answer");
chooseabcd = _getch();
cout << chooseabcd << endl;
if(chooseabcd == 'a'||chooseabcd == 'A')
```

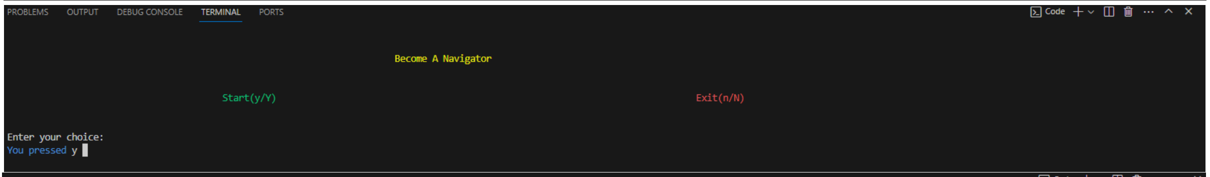
```
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'b'||chooseabcd == 'B')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'c'||chooseabcd == 'C')
{
typer("Wrong Direction");
Sleep(3000);
Wrong_direction++;
system("cls");
if(Wrong_direction>=2)
{
goto End;
}
}
else if(chooseabcd == 'd'||chooseabcd == 'D')
{
typer("Correct Direction");
typer("Taking U-Turn your Back at where you strated");
Sleep(3000);
system("cls");
goto Officeloc;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto Southturngoto;
}
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto Officeloc;
}
typer("Congratulations you have passed you first job \n LEVEL PASSED");
```

```
End:
if(Wrong_direction>=2)
{
system("cls");
typer("You have given 2 wrong directions" );
typer("You are removed from the job!!" );
Sleep(3000);
}
system("cls");
typer("GAME OVER\n\nDo you want to Replay the game or EXIT");
typer("Press \nR for REPLAY\nE for EXIT");
replayorexit = _getch();
replayorexitgoto:
if(replayorexit == 'r'||replayorexit == 'R')
{
goto Start;
}
else if(replayorexit == 'e'||replayorexit == 'E')
{
Exit:
system("cls");
typer("\nExiting in 5 secounds");
Sleep(1000);
system("cls");
cout << "\nExiting in 4 secounds";
Sleep(1000);
system("cls");
cout << "\nExiting in 3 secounds";
Sleep(1000);
system("cls");
cout << "\nExiting in 2 secounds";
Sleep(1000);
system("cls");
cout << "\nExiting in 1 secounds";
Sleep(1000);
system("cls");
return 0;
}
else
{
typer("\nWrong Input... Try again...");
Sleep(1000);
system("cls");
goto replayorexitgoto;
}
}
```

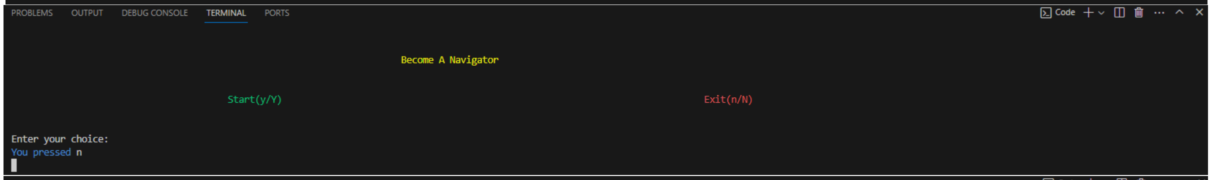
Output:



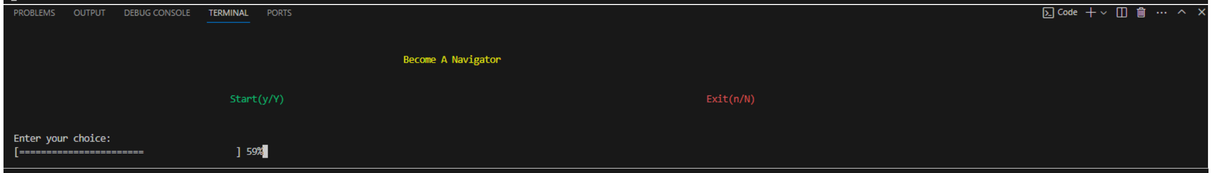
```
                                        Become A Navigator

                    Start(y/Y)                                                  Exit(n/N)

Enter your choice:
```



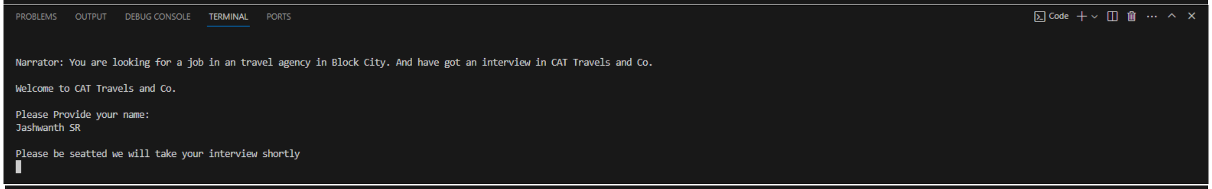
```
                                        Become A Navigator

                    Start(y/Y)                                                  Exit(n/N)

Enter your choice:
You pressed y
```



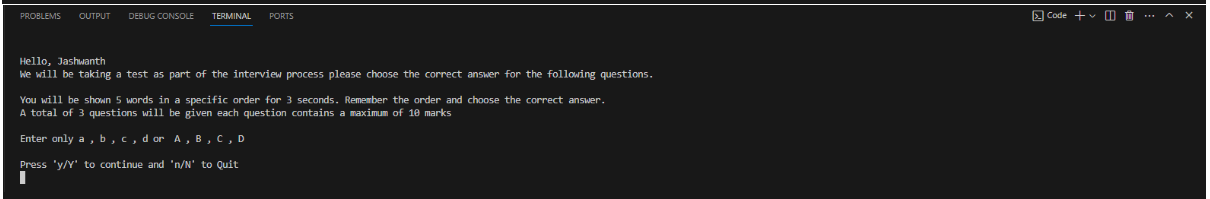
```
                                        Become A Navigator

                    Start(y/Y)                                                  Exit(n/N)

Enter your choice:
You pressed n
```



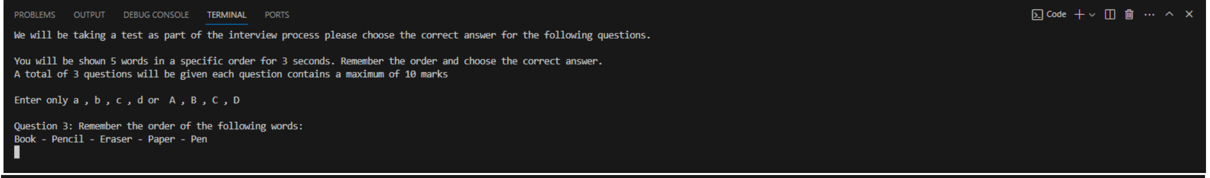
```
                                        Become A Navigator

                    Start(y/Y)                                                  Exit(n/N)

Enter your choice:
[======================                    ] 59%
```



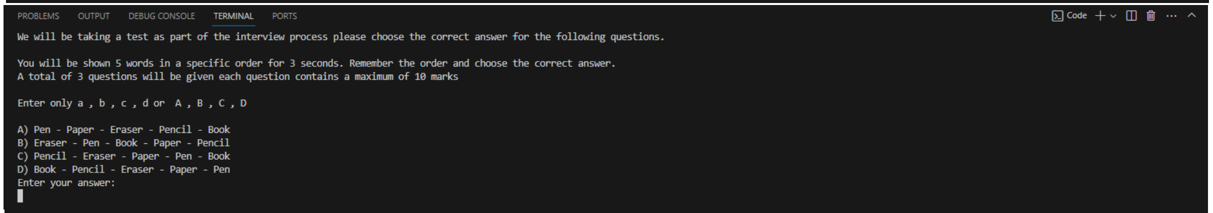
```
Narrator: You are looking for a job in an travel agency in Block City. And have got an interview in CAT Travels and Co.

Welcome to CAT Travels and Co.

Please Provide your name:
Jashwanth SR

Please be seatted we will take your interview shortly
```



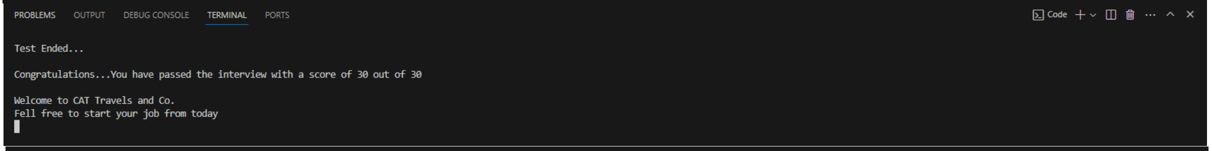
```
 Hello, Jashwanth
 We will be taking a test as part of the interview process please choose the correct answer for the following questions.

 You will be shown 5 words in a specific order for 3 seconds. Remember the order and choose the correct answer.
 A total of 3 questions will be given each question contains a maximum of 10 marks

 Enter only a , b , c , d or  A , B , C , D

 Press 'y/Y' to continue and 'n/N' to Quit
```



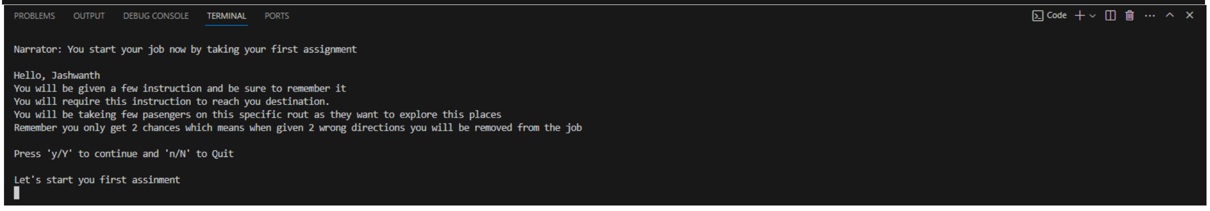
```
We will be taking a test as part of the interview process please choose the correct answer for the following questions.

You will be shown 5 words in a specific order for 3 seconds. Remember the order and choose the correct answer.
A total of 3 questions will be given each question contains a maximum of 10 marks

Enter only a , b , c , d or  A , B , C , D

Question 3: Remember the order of the following words:
Book - Pencil - Eraser - Paper - Pen
```



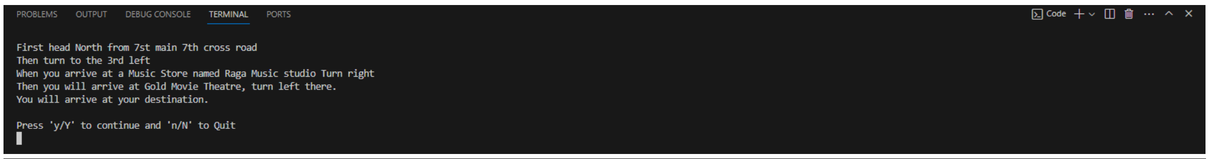
```
We will be taking a test as part of the interview process please choose the correct answer for the following questions.

You will be shown 5 words in a specific order for 3 seconds. Remember the order and choose the correct answer.
A total of 3 questions will be given each question contains a maximum of 10 marks

Enter only a , b , c , d or  A , B , C , D

A) Pen - Paper - Eraser - Pencil - Book
B) Eraser - Pen - Book - Paper - Pencil
C) Pencil - Eraser - Paper - Pen - Book
D) Book - Pencil - Eraser - Paper - Pen
Enter your answer:
```



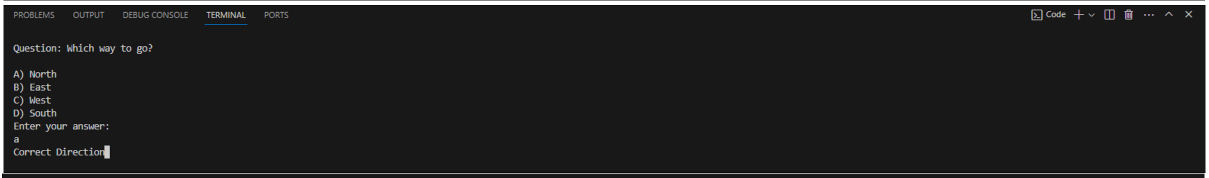
```
Test Ended...

Congratulations...You have passed the interview with a score of 30 out of 30

Welcome to CAT Travels and Co.
Fell free to start your job from today
```



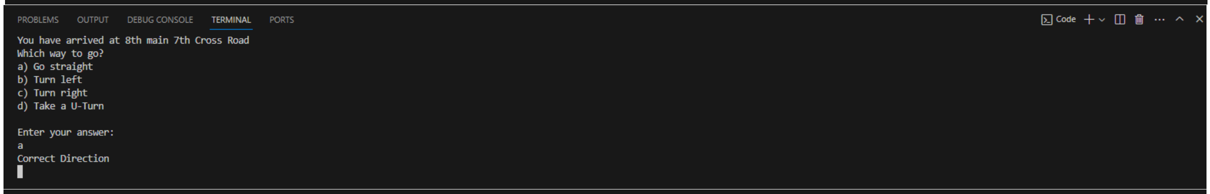
```
Narrator: You start your job now by taking your first assignment

Hello, Jashwanth
You will be given a few instruction and be sure to remember it
You will require this instruction to reach you destination.
You will be takeing few pasengers on this specific rout as they want to explore this places
Remember you only get 2 chances which means when given 2 wrong directions you will be removed from the job

Press 'y/Y' to continue and 'n/N' to Quit

Let's start you first assinment
```



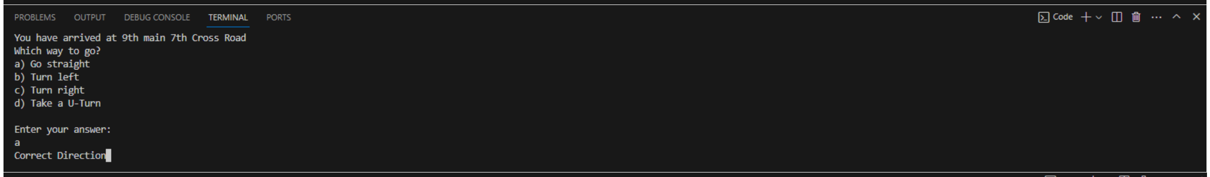
```
First head North from 7st main 7th cross road
Then turn to the 3rd left
When you arrive at a Music Store named Raga Music studio Turn right
Then you will arrive at Gold Movie Theatre, turn left there.
You will arrive at your destination.

Press 'y/Y' to continue and 'n/N' to Quit
```



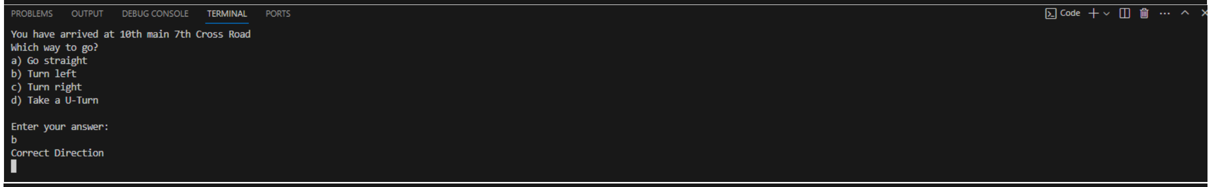
```
Question: Which way to go?

A) North
B) East
C) West
D) South
Enter your answer:
a
Correct Direction
```



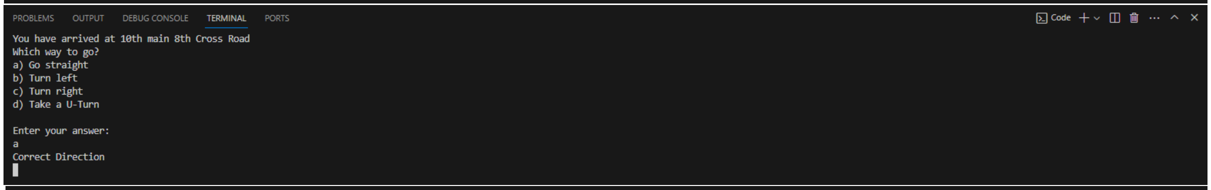
```
You have arrived at 8th main 7th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
a
Correct Direction
```



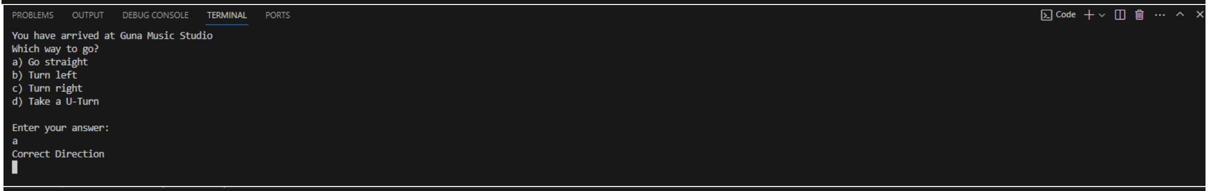
```
You have arrived at 9th main 7th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
a
Correct Direction
```



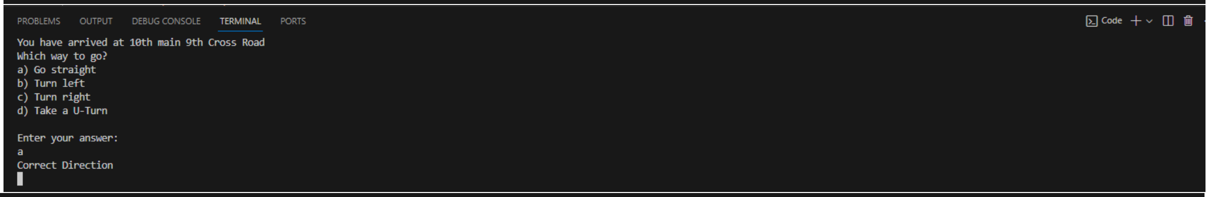
```
You have arrived at 10th main 7th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
b
Correct Direction
```



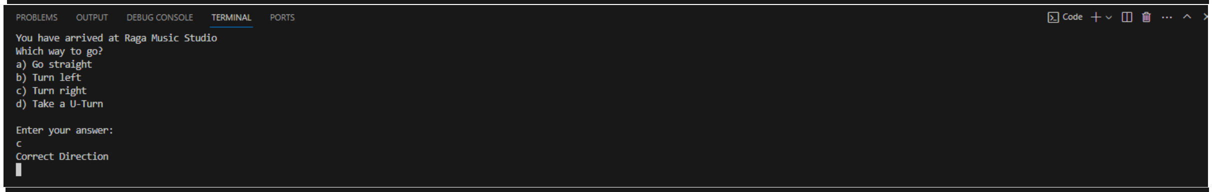
```
You have arrived at 10th main 8th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
a
Correct Direction
```



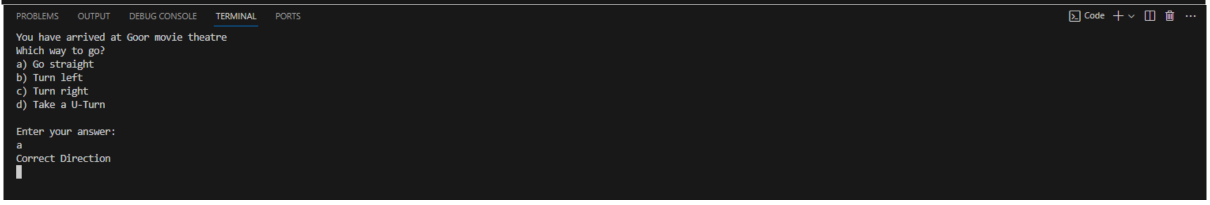
```
You have arrived at Guna Music Studio
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
a
Correct Direction
```



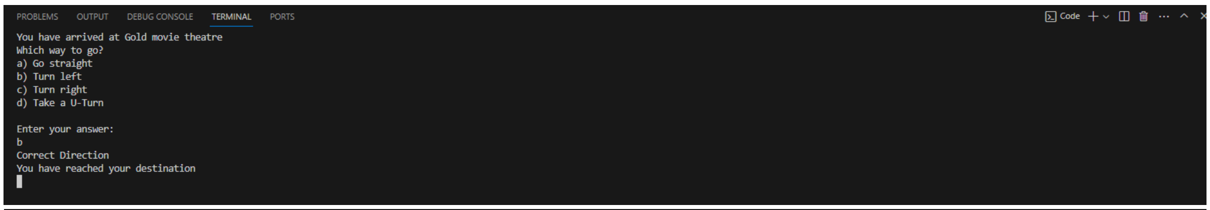
```
You have arrived at 10th main 9th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
a
Correct Direction
```



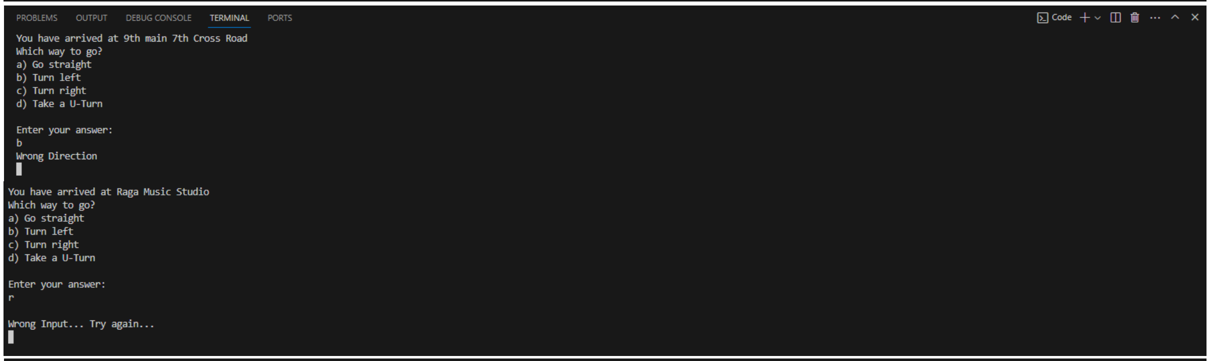
```
You have arrived at Raga Music Studio
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
c
Correct Direction
```



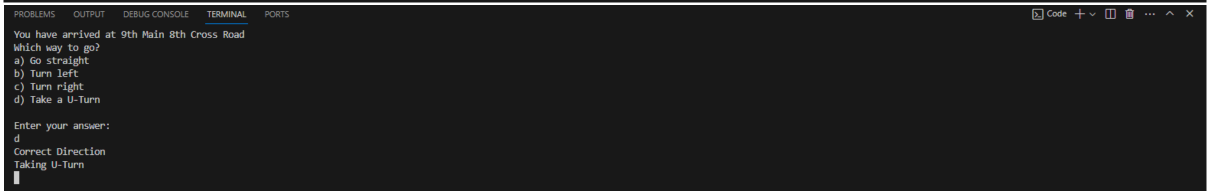
```
You have arrived at Goor movie theatre
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
a
Correct Direction
```



```
You have arrived at Gold movie theatre
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
b
Correct Direction
You have reached your destination
```



```
You have arrived at 9th main 7th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
b
Wrong Direction

You have arrived at Raga Music Studio
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
r

Wrong Input... Try again...
```



```
You have arrived at 9th Main 8th Cross Road
Which way to go?
a) Go straight
b) Turn left
c) Turn right
d) Take a U-Turn

Enter your answer:
d
Correct Direction
Taking U-Turn
```

Problem Faced:

1. The main problem faced was the game looping as there are many places where the program must run from back again.

2. There were a few bugs as I have created the game based on an idea of a map being imagined by the player.

3. The code was too long and confusing sometimes.

4. C++ was a new language for me.

5. Game was very normal which does not suite my style of programming.

6. The concept chosen was a bit tricky to understand.

How I overcame the challenges:

1. Use of ‘goto statement’ made programming easier.

2. By changing the way of my programming style and learning more functions in C++, the bug fixing part became easy.

3. Using the VS Code software navigating through the code was easy.

4. Using few online tutorials and my teachers help I learned to program in C++ quickly.

5. By changing the output style and adding colours the game looked more interesting to make.

6. By making few changes and add-on to the concept made it more interesting to make.

Things to Improve:

1. Learning C++ in more detailed will help and improve my skills.
2. Working on more Text-Based games can help learn the core of programming in Game Development.
3. Using of public Libraries can make the game look neater.
4. Building the game in a game environment will make the game even smoother.
5. Playing more text-based game can help in understanding them better.

Things I Learned:

1. C++ programming language was the first thing I learned.
2. Learnt about Text-based games and how they are made.
3. Learned how to work with Structures, Functions, and other Libraries.
4. Learned to make a console based game where the player interacts with only text and the rest is up to the players imagination.