# Report

Name: Huzaifa Muhammad Siddique
Student ID: **22050141041**

This report is prepared on a “Dungeon Exploration Text-Based” Game made in the C programming language. Here are the contents of this report:

- Introduction
- Gameplay
- Design and Implementation
- Commands
- Rooms
- Rooms Map
- Functions
- Structs
- Items
- Creatures
- Weapons

# Introduction

In this game we have a total of 9 rooms, 25 items, 7 creatures and 6 weapons. Weapons are actually part of items but they are treated differently. The player starts the game in a room called “Arena”. The goal of the game is to kill all of the creatures in all rooms. The player can collect items, fight creatures, equip weapons, go to the shop and buy and sell items, and look at room details. The player can also move around between the rooms.

# Gameplay

The player has attributes of health, strength and money. The player also has an inventory where he can store items. The maximum number of items he can store in the inventory are 20. The player starts in the “Arena” room and can move to another room using the `move` command. With the `move` keyword, the user must also specify the direction. If there is a room in that direction, the user is moved to that room, otherwise a message is displayed telling the user that there is a wall in that direction. The player can also fight the creatures in the room. And the fight mechanism is very simple. First the player hits the creature with its strength, then the creature hits the player with its strength but there is a chance of random increase or decrease in the damage the creature can make. This goes on until the health of either the creature or the player gets to or below 0. To fight a creature, the player has to use the command `attack` followed by the name of the creature. If the name of the creature is wrong or if the creature is not present in the current room, an appropriate message is displayed. The player can also examine the surrounding of the current room by just entering the `look` command. This shows the description, creatures and items of the current room. The player can pick up items from the current room. If a wrong item is entered or an item that is not present in the room is entered, an appropriate message is displayed. For picking up items the player has to use the `pickup` command followed by the name of the item the player wishes to pick up. The player can also save the current state of a game to a text file by using the `save` command followed by a file path to the text file. Likewise, the user can also load a game state and continue playing it. For this, the user has to use the `load` command followed by the path of a text file from where the game state must be loaded. The player can also list all of the saved games by using the `list` command. The player can also look at all of the items that he possesses. For this the player can use the `inventory` command. This displays all of the items that are present in the inventory of the player. The player can also use potion if he has it in his inventory. This potion increases the health of the player. For this, the user has to type the `use Potion` command and if there is a potion in the inventory of the player, his health will be increased. Otherwise, an error message will be displayed. The player can also equip weapons. Weapons

either increase the health of the player or his strength. It depends upon what type of weapon it is. For equipping a weapon, the user has to type the command `equip` followed by the name of the weapon. If this weapon is present in the inventory of the user, it will be successfully equipped. Otherwise, an error message will be displayed. The player can also use the `player` command to see his health and strength. The user can also visit the shop using the `shop` command. Once the player is in the shop, there are several other commands that he can use and these commands *only* work in the shop. If the user wants to buy something, the user can type in the command `buy` followed by what he wants to buy. In this game, the user can only buy health and strength. The user can also sell items. And for this the user has to use the `sell` command followed by the name of the item that the user wants to sell. If that item is present in the player's inventory, it is removed from there and an appropriate amount of money is added to the player's money. If the item is not present in the player's inventory, an error message is displayed. The player can exit the shop using the `exit shop` command. At last, if the user wants to exit the game, he can simply do this by using the `exit` command.

# Design and Implementation

The game has been implemented using structs, functions, loops and if statements. The design of the game is very simple. A struct has been created for each of the entities in the game i.e. Player, Creature and Room. Then all of the rooms are stored in one array and all of the creatures are stored in one array. All of the data related to the game is stored in files. That is why there are a lot of files. And using a design like this makes making changes to the game very simple. Like if I want to change the health and strength of a creature, I can simply update its file. If I want to add items or creatures to a room or if I want to remove items or creatures from a room, I just need to update the file of that room. For implementing most of the features, functions have been created and used. There are a total of 25 functions used in the game. To make the design simple and easy, all of the code has been written in one file. For the design of the flow of the game, an infinite loop has been

used that keeps on taking commands from the user and keeps on executing them. The loop breaks on three conditions:

1. The player uses the “exit” command to close the program.
2. The player is defeated by one of the enemies.
3. The player defeats all of the enemies and wins the game.

There is an infinite loop used inside the shop as well. This is because it has been attempted to create a similar mechanism of taking commands from the user and executing them. This loop only breaks when the player exits from the shop. Dynamic memory allocation is used for strings when they are read from the file. For this, built-in string methods have been used.

# Commands

## Game Commands

`look`

Lists the description of the current room as well as the creatures and items in it.

`inventory`

Displays the items in the inventory of the player. It shows all the items that the player possesses.

`move up`

Moves the player to the room upwards if there is a room.

`move down`

Moves the player to the room downwards if there is a room.

`move right`

Moves the player to the room to the right if there is a room.

`move left`

Moves the player to the room to the left if there is a room.

`pickup <item>`
Picks up the specified item from the current room and adds it to the inventory if it is present in the current room.

`attack <creature>`
Fights the specified creature if it is present in the room. The result of the fight is displayed.

`equip <weapon>`
Equips a weapon if it is present in the inventory of the user. Equipping weapons increases the health or the strength of the player depending upon the type of weapon.

`use Potion`
Increases the health of the user if a Potion is present in the inventory of the player.

`player`
Displays the health and the strength of the user.

`exit`
Closes the game.

`save <filepath>`
Saves the current game state to the text file in the specified file path.

`load <filepath>`
Loads the game state saved in the text file specified on the file path.

`list`
Displays all of the file paths where games are saved.

`shop`
Opens the shop. Player now goes to the shop and only shop-related commands can work.

## Shop Commands

`buy <item>`

The player can either buy health or strength if he has enough money.

`sell <item>`

Allows the player to sell an item if the item is present in the player's inventory. Removes the item from the player’s inventory and adds the appropriate amount to the player's money.

`exit shop`

Closes the shop. The player comes out of the shop and now he can use other game commands.

# Rooms

There are 9 rooms in the game. Here is a list of the Rooms:

1. Arena
2. Chamber
3. Enterance Hall
4. Guardroom
5. Library of the Lost
6. Potion Laboratory
7. Spider Nest
8. Treasure Vault
9. Trophy Room

# Rooms Map

This is a map of how all of the rooms are aligned to each other. The borders represent walls.

| | | |
|---|---|---|
| **Potion Laboratory** | **Chamber** | **Spider Nest** |
| **Guardroom** | **Arena** | **Treasure Vault** |
| **Library of the Lost** | **Enterance Hall** | **Trophy Room** |

# Functions

Following are all of the functions that have been used throughout the program. Explanation of what each function does is also given.

`startingMessage()`
Used to display the starting message at the start of the game.

`loadRoomStatsFromFile(struct Room, FILE)`
Used to read data from a room file and store it in a room struct. All data is read and stored except for the creatures' data.

`loadCreaturesInRoom(struct Room, FILE)`
Used to read the creatures' data for a room from a file and load it into the room struct.

`loadCreatureStatsFromFile(struct Creature, FILE)`
Used to read data from a creature file and store it in a creature struct.

`alignRooms(struct Room[])`
Used to make a map-like structure of the rooms by deciding which room lies in which direction of other rooms.

`getRoomByName(char*, struct Room[])`
Uses the name of a room and returns the struct of that room.

`getCreatureByName(char*, struct Creature[])`
Uses the name of a creature and returns the struct of that creature.

`displayAllItems(struct Room)`
prints out all of the items in a room.

`displayAllCreatures(struct Room)`
prints out all of the creatures in a room.

`itemInRoom(char*, struct Room)`
checks if an item is present in a room.

`creatureInRoom(char*, struct Room)`
checks if a creature is present in a room.

`addToInventory(char*)`
adds an item to the file where the inventory items are stored.

`battle(struct Player, struct Creature)`
This is used to fight a creature. If the player wins, true is returned otherwise false is returned.

`removeItemFromRoom(struct Room, char*)`
This removes an item from the specified room if the item is present in the room.

`removeCreatureFromRoom(struct Room, char*)`
This removes a creature from the specified room if the creature is present in the room.

`potionInInventory(char*[])`
This checks if Potion is present in the inventory.

`overwriteInventory(char*[], FILE)`
This updates the contents of the file where the inventory items are stored.

`itemInInventory(char*, char*[])`
This checks if a specific item is present in the inventory.

`countItemsInInventory(char*[])`
This returns the number of items that are present in the inventory.

`getItemPrice(char*, FILE)`
This reads the price of an item from the file and returns it as an integer.

`updateRoom(struct Room, struct Room[])`
This is a very important function. It updates the state of the currentRoom and stores it in the array that contains all of the rooms.

`saveGame(struct Player, struct Room[], struct Room, char*)`
saves a game state to a text file.

`loadGame(struct Player, struct Room[], struct Room, char*)`
loads a game state from a text file.

`listSavedGames()`
prints out the paths of all of the files where saved games are stored.

`calculateRemainingCreatures(struct Room[], int)`
calculates the total creatures left in all of the rooms and returns it as an integer.

# Structs

There are three structs used in the game. Here is a list:

1. Player
2. Room
3. Creature

Player struct for the player who is playing the game. Creature struct for each one of the creatures. Room struct for each room.

# Items

There are 25 Items in the game. Here is a list of them:

| | | | | | | |
|---|---|---|---|---|---|---|
| Glasses | Keyboard | Bronze | Bow | Mirror | Diamond | Bottle |
| Gemstone | Gold | Gun | Potion | Helmet | Knife | Paper |
| Phoenix | Rope | Shield | Silver | Steel | Sword | Book |
| Treasure | Trophy | Tyre | Wood | | | |

# Creatures

There are 7 creatures in the game. Here is a list of them:

| | | | | | |
|---|---|---|---|---|---|
| Barbarian | Dragon | Goblin | Spider | Troll | Warrior |
| Zombie | | | | | |

# Weapons

There are 6 weapons in the game. Here is a list of them:

| | | | | | |
|---|---|---|---|---|---|
| Gun | Sword | Knife | Rope | Bow | Shield |