version: 1.0 last updated: 2025-03-19 09:00

CS Dungeon

Overview

Your task in assignment 2 is to create a dungeon crawler game, where you will get to design a map of connected dungeons filled with monsters, items, and a final boss at the end to defeat! You will get to play as either a Fighter or a Wizard, using your special skills and stats to defeat these monsters and beat the game!

The game consists of a setup phase, where you add dungeons and items, and a gameplay phase, where you as the player will get to move between dungeons, fight monsters and collect items that can make you even stronger!

You can read about the history of dungeon crawler games here.

Assignment Structure

This assignment will test your ability to create, use, manipulate and solve problems using linked lists. To do this, you will be implementing a dungeon crawler game, where the dungeons are represented as a linked list, stored within a map. Each dungeon contains a list of items. The map also contains a player struct, which also contains a list of items.

We have defined some structs in the provided code to get you started. You may add fields to any of the structs if you wish.

NOTE:

There are 5 structs used in this assignment. You do not need to know everything from the beginning, each stage of the assignment will walk you through what you need to know. The easiest way to understand these structs is to get stuck into **Stage 1.1** where you get to create some!

– Structs

`struct map`

- Purpose: To store all the information about the dungeon map. It contains the list of dungeons within the map, the player, and the number of points required to win the game. The `entrance` field is a pointer to the first dungeon in the list of dungeons.
- Defined in `cs_dungeon.h` .

`struct dungeon`

- Purpose: To store all the information about a single dungeon. This will form a linked list of dungeons by pointing to the next one (or `NULL`).
- Defined in `cs_dungeon.c` .

`struct item`

- Purpose: To store all the information about a single item. This will form a linked list of items by pointing to the next one (or `NULL`). Items can be stored in a dungeon and in the player's inventory.

- Defined in `cs_dungeon.c` .

`struct player`

- Purpose: To store all the information about the player.
- Defined in `cs_dungeon.c` .

`struct boss`

- Purpose: To store all the information about the final boss.
- Defined in `cs_dungeon.c` .

The following enum definitions are also provided for you. You can create your own enums if you would like, but you should not modify the provided enums.

– Enums

`enum monster_type`

- Purpose: To represent the different types of monsters that can be encountered in the dungeons.
- Defined in `cs_dungeon.h` .

`enum item_type`

- Purpose: To represent the different types of items that can be found in the dungeons.
- Defined in `cs_dungeon.h` .

HINT:

Remember to initialise every field inside the structs when creating them (not just the fields you are using at that moment).

Getting Started

There are a few steps to getting started with CS Dungeon.

1. Create a new folder for your assignment work and move into it. You can follow the commands below to link and copy the files.

```
$ mkdir ass2
$ cd ass2
```

2. There are 3 files in this assignment. Run the following command below to download all 3, which will link the header file and the main file. This means that if we make any changes to `cs_dungeon.h` or `main.c` , you will not have to download the latest version as yours will already be linked.

```
$ 1091 fetch-activity cs_dungeon
```

3. Run `1091 autotest cs_dungeon` to make sure you have correctly downloaded the file.

```
$ 1091 autotest cs_dungeon
```

WARNING:

When running the autotest on the starter code (with no modifications), it is expected to see failed tests.

4. Read through the rest of the introductory specification and **Stage 1**.

Starter Code

This assignment utilises a multi-file system. There are three files we use in CS Dungeon:

- **Main File (`main.c`):** This file handles **all input scanning and error handling** for you. It also tests your code in `cs_dungeon.c` and contains the `main` function. You don't need to modify or fully understand this file, but if you're curious about how this assignment works, feel free to take a look. **You cannot change `main.c` .**
- **Header File (`cs_dungeon.h`):** contains defined constants that you can use and function prototypes. It also contains header comments that explain what functions should do and their inputs and outputs. *If you are confused about what a function should do, read the header file and the corresponding specification.* **You cannot change `cs_dungeon.h` .**
- **Implementation File (`cs_dungeon.c`):** contains stubs of functions for you to implement. This file does not contain a `main` function, so you will need to compile it alongside `main.c` . *This is the only file you may change. You do not need to use `scanf` or `fgets` anywhere.*

NOTE:

Function Stub: A temporary substitute for yet-to-be implemented code. It is a placeholder function that shows what the function will look like, but does nothing currently.

The implementation file `cs_dungeon.c` contains some provided functions to help simplify some stages of this assignment. These functions have been fully implemented for you and should not need to be modified to complete this assignment.

These provided functions will be explained in the relevant stages of this assignment. **Please read the function comments and the specification as we will suggest certain provided functions for you to use.**

WARNING:

Do not change the return type or parameter amount and type of the provided function stubs. `main.c` depends on these types and your code may not pass the autotests otherwise, as when testing we will compile your submitted `cs_dungeon.c` with the supplied `main.c` .

NOTE:

If you wish to create your own helper functions, you can put the function prototypes at the top of `cs_dungeon.c` and implement it later in the file. You should place your function comment just above the function definition.

How to Compile CS Dungeon

– Compiling CS Dungeon

To compile you should compile `cs_dungeon.c` alongside `main.c` . This will allow you to run the program yourself and test the functions you have written in `cs_dungeon.c` . Autotests have been written to compile your `cs_dungeon.c` with the provided `main.c` .

To compile your code, use the following command:

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
```

Once your code is compiled, you can run it with the following command:

```
$ ./cs_dungeon
```

To autotest your code, use the following command:

```
$ 1091 autotest cs_dungeon
```

Reference Implementation

To help you understand the expected behaviour of CS Dungeon, we have provided a reference implementation. If you have any questions about the behaviour of your assignment, you can check and compare yours to the reference implementation.

To run the reference implementation, use the following command:

```
$ 1091 cs_dungeon
```

− Example Usage

Once you have followed the setup prompts, you might want to start by running the `?` command, whether you are in the setup phase or the gameplay phase.

When in the setup phase, the `?` command will display what you can add to the map:

```
 $  1091 cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: ?
=======================[  1091 Dungeon  ]=======================
      =============[    Setup: Usage Info   ]===============
  ?
    Show setup usage info
  q
    Exit setup stage
  a [dungeon name] [monster type] [num_monsters]
    Append a dungeon to the end of the map's list of dungeons
  p
    Prints the map's list of dungeons
  s
    Shows the player's current stats
  i [position] [dungeon name] [monster type] [num_monsters]
    Inserts a dungeon to the specified position in the map
  t [dungeon position] [item type] [points]
    Inserts an item into the specified dungeon's list of items
================================================================

Enter Command: Ctrl-D
Thanks for playing!
```

When in the gameplay phase, the `?` command will show you what actions the player can take:

```
 $  1091 cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------



Enter Command: a beach 1 1
beach has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 1
The final boss has been added to faerun!
Map of faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. beach
Boss: Present
Monster: Slime
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------



Enter Command:?
=======================[  1091 Dungeon  ]=======================
      =============[  Gameplay: Usage Info  ]===============
  ?
    Show gameplay usage info
  q
    Exit gameplay
  p
    Prints the map's list of dungeons
  s
    Shows the player's current stats
  d
    Prints the current dungeon's details
  >
    Move to the next dungeon in the map
  <
    Move to the previous dungeon in the map
  !
    Physically attack monsters in current dungeon
  #
    Magically attack monsters in current dungeon
  P
    Activate the player's class power
  c [item number]
    Collect the specified item from current dungeon
  u [item number]
    Use the specified item from the player's inventory
  T
    Teleport to the furthest dungeon
  b
    Fight the boss in the current dungeon
================================================================

Enter Command: Ctrl-D
Thanks for playing!
```

The easiest way to understand how this assignment works is to play a game yourself! Below is example input you can try by using the reference solution.

– Example Game

```
Faerun
25
Minsc
Fighter
a cavern 1 3
a castle 2 2
a graveyard 3 3
t 1 0 5
t 1 0 3
t 2 1 5
t 2 3 5
t 2 3 5
q
1
```

Allowed C Features

In this assignment, there are no restrictions on C Features, except for those in the Style Guide. If you choose to disregard this advice, you must still follow the Style Guide.

You also may be unable to get help from course staff if you use features not taught in DPST1091/CPTG1391. Features that the Style Guide identifies as illegal will result in a penalty during marking. You can find the style marking rubric above. **Please note that this assignment must be completed using only Linked Lists . Do not use arrays in this assignment. If you use arrays instead of linked lists you will receive a 0 for performance in this assignment**.

Banned C Features

In this assignment, **you cannot use arrays for the list of dungeons nor the lists of items** and cannot use the features explicitly banned in the Style Guide. If you use arrays in this assignment for the linked list of dungeons or the linked lists of items you will receive a **0** for performance in this assignment.

FAQ

– FAQ

Q: Can I edit the given starter code functions/structs/enums?

You can only edit `cs_dungeon.c` .You **cannot** edit anything in `main.c` or `cs_dungeon.h` .

You **cannot** edit any of the parameters or names for the provided functions.

Q: Can I use X other C feature

A: For everything not taught in the course, check the style guide. If it says "Avoid", then we may take style marks off if its not used correctly. If it says "Don't Use" then we will take style marks off (see the style marking rubric).

Game Structure

This game consists of a setup phase and a gameplay phase.

You will be implementing both the setup phase and gameplay phase throughout this assignment, adding more features to both as you progress. By the end of **stage 1.4****you will have implemented parts of both setup and gameplay enough to play a very basic game.

The game is ended either by the user entering `Ctrl-D` , the win condition being met, or the player running out of health points. The program can also be ended in the setup phase with `Ctrl-D` or `q` .

Your Tasks

This assignment consists of four stages. Each stage builds on the work of the previous stage, and each stage has a higher complexity than its predecessor. You should complete the stages in order.

NOTE:

You can assume that your program will **NEVER** be given:

- A non-existent command.
- Command arguments that are not of the right type.
- An incorrect number of arguments for the specific command.

Additionally, commands will always start with a `char` . **All scanning and most printing will be handled for you in `main.c` , so you should not need to worry about the above.**

Stage 1 ●○○ Stage 2 ●○○ Stage 3 ●●○ Stage 4 ●●● Extension Tools

Stage 2

In **Stage 2** of this assignment, you will be using more advanced linked list concepts to add and move nodes within a linked list and perform operations using the values stored inside those nodes.

Specifically, this will include:

- Inserting a dungeon into the map at a specified position.
- Print details about the dungeon the player is currently in.
- Move the player between dungeons.
- Fight monsters (not the final boss).
- Implement class-specific powers.

Stage 2.1 - Inserting Dungeons

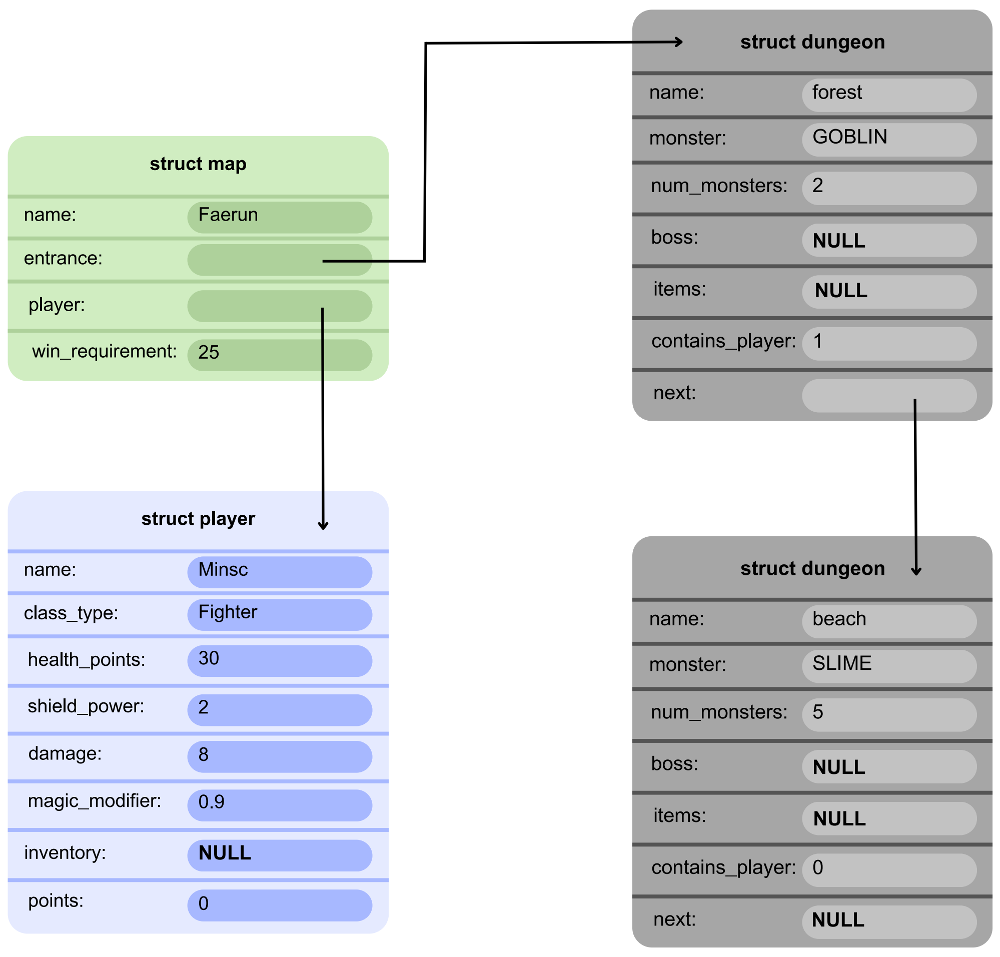
Currently, we have a way of appending a dungeon to the end of the map's list of dungeons, but there's no way to insert a dungeon anywhere else in your list.

That's why you will be implementing the Insert Dungeon command for **Stage 2.1**, which will insert a new dungeon at the specified position, rather than at the end of the map's list of.

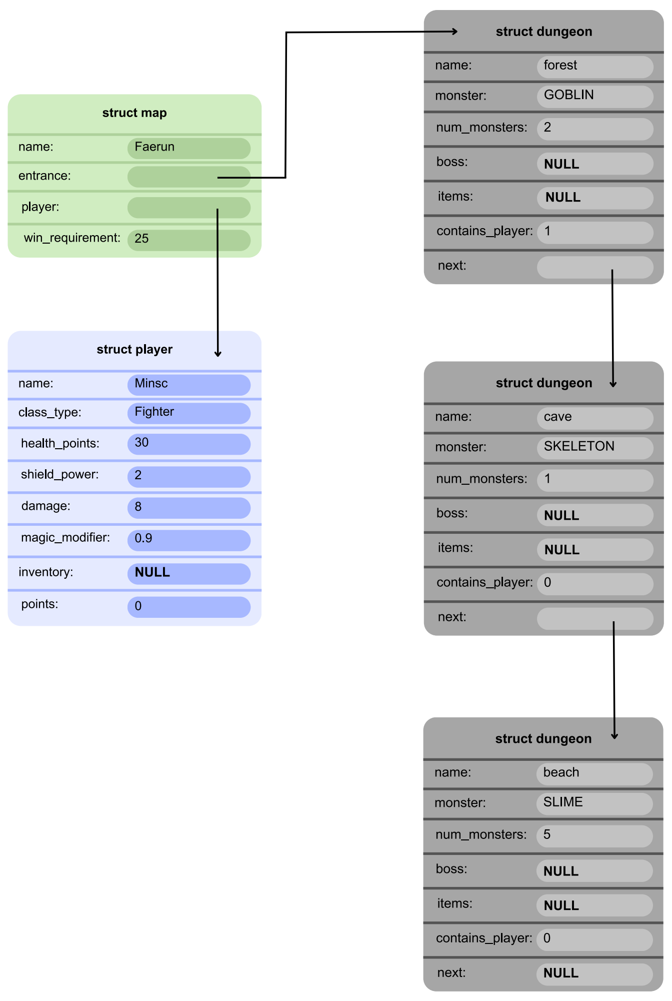
```
Enter Command: i [position] [name] [monster] [num_monsters]
```

Inserting a new dungeon should look something like this:

Before Inserting:



After Inserting:



For this stage, you will need to implement the function `insert_dungeon` in `cs_dungeon.c` .

For both of these functions, your program should create a new `struct dungeon` and then insert it into the map's list of dungeons at the given `position` . **The list is indexed from 1, meaning the head of the list is position 1. The `position` given means that the new dungeon should take that position in the list**, e.g. if the position was `1` , the new dungeon should become the new `entrance` of the map's list of dungeons.

This function returns an `int` - this value indicates whether or not the inserting was successful or not. As we handled error checking in **Stage 1.5**, you will now need to check if any of the arguments provided are invalid. If they are, you will need to return the appropriate constant, defined in `cs_dungeon.h` .

`main.c` will then handle the printing accordingly.

If a dungeon is invalid, it should not be inserted to the list of dungeons in the map.

Clarifications

- Your program should also check if `position` is invalid. `position` is invalid if it is less than 1.
- Your program should handle the same errors as **stage 1.5**. Invalid position should be checked first, then the rest of the errors in the same order as **Stage 1.5**.
- The list is indexed from 1, meaning the head of the list (`entrance`) is position 1. The `position` given means that the new dungeon should take that position in the list, and should point to the dungeon that previously held that position.
- If the given `position` is 1, the new dungeon should become the new entrance of the map.
- As stated in **Stage 1.2**, the player should start in the `entrance` dungeon of the map, so you will need to update this as well.
- If the given `position` is larger than the length of the list, the new dungeon should be added to the tail of the list.

Examples

– Example 2.1.1: Inserting at the beginning of the list

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: i 1 forest 3 2
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 1
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: Present
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: Ctrl-D
Thanks for playing!
```

– Example 2.1.2: Inserting at the end of the list

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a beach 1 2
beach has been added as a dungeon to the map!

Enter Command: a castle 2 2
castle has been added as a dungeon to the map!

Enter Command: i 3 forest 3 2
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 1
The final boss has been added to Fearun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. beach
Boss: None
Monster: Slime
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. castle
Boss: None
Monster: Goblin
Empty

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

3. forest
Boss: Present
Monster: Skeleton
Empty

|^|^|^|^|^|   |^|^|^|^|^|

--------Gameplay Phase--------

Enter Command: Ctrl-D
Thanks for playing!
```

– Example 2.1.3: Inserting in the middle of the list

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a beach 1 2
beach has been added as a dungeon to the map!
castle Command: a castle 2 2
castle has been added as a dungeon to the map!

Enter Command: i 2 forest 3 2
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 1
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. beach
Boss: None
Monster: Slime
Misc is here

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. forest
Boss: None
Monster: Skeleton
Empty

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

3. castle
Boss: Present
Monster: Goblin
Empty

|^|^|^|^|^|   |^|^|^|^|^|

--------Gameplay Phase--------

Enter Command: Ctrl-D
Thanks for playing!
```

– Example 2.1.4: Inserting at invalid position

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter

----------Setup Phase----------

Enter Command: i 0 forest 3 2
ERROR: Invalid position. Position must be 1 or greater.

Enter Command: Ctrl-D
Thanks for playing!
```

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 02_01 cs_dungeon
```

Stage 2.2 - Print Current Dungeon Details

We need a way to see all the details about the dungeon the player is currently in, specifically:

- How many monsters are in the dungeon
- Boss details (if there is a boss)
- All items in the dungeon (at this stage there should be no items)

```
Enter Command: d
```

There is one function in `cs_dungeon.c` that you will have to implement for **Stage 2.2**, `print_dungeon` .

Another helper function has been provided to you, in addition to `print_no_items` and `print_item` from **Stage 1.6**. You do not need to use `print_item` until items are added in **Stage 3**.

- `print_detail_dungeon` prints all the specific details about a dungeon, as listed above.

Examples

– Example 2.2.1: Checking dungeon details, one dungeon

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a forest 3 2
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: Present
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 2 Skeletons
The boss is in this dungeon
        Health Points: 35
        Damage: 10
        Points: 20
        Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: Ctrl-D
Thanks for playing!
```

– Example 2.2.2: Checking dungeon details, two dungeons

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a forest 3 2
forest has been added as a dungeon to the map!

Enter Command: a cave 1 1
cave has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: None
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. cave
Boss: Present
Monster: Slime
Empty

|^|^|^|^|^|   |^|^|^|^|^|

--------Gameplay Phase--------


Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 2 Skeletons
No boss in this dungeon
The dungeon forest has the following items:
No Items

Enter Command: Ctrl-D
Thanks for playing!
```

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 02_02 cs_dungeon
```

Stage 2.3 - Move Player Between Dungeons

We've created a map with a list of dungeons, now we want to be able to actually move the player through the dungeons!

Moving forward to the next dungeon:

```
Enter Command: >
```

Moving backward to the previous dungeon:

```
Enter Command: <
```

There is one function in `cs_dungeon.c` that you will have to implement for **Stage 2.3**. `move_player` , which will move the player from the dungeon they are currently in, to an adjacent dungeon, depending on whether `>` or `<` was inputted.

This function returns an `int` - this value indicates whether or not the moving was successful or not. Return `VALID` if there is a dungeon to move into, and `INVALID` if not.

`main.c` will then handle the printing accordingly.

Clarifications

- If there is no dungeon the player can move into, the player should not move.

Examples

– Example 2.3.1: Moving between dungeons

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a forest 3 2
forest has been added as a dungeon to the map!

Enter Command: a cave 1 1
cave has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: None
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. cave
Boss: Present
Monster: Slime
Empty

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: >
Moved into the next dungeon

Enter Command: p
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: None
Monster: Skeleton
Empty

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. cave
Boss: Present
Monster: Slime
```

```
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|

Enter Command: <
Moved into the previous dungeon

Enter Command: p
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: None
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. cave
Boss: Present
Monster: Slime
Empty

|^|^|^|^|^|   |^|^|^|^|^|

Enter Command: <
ERROR: Invalid move. There must be a dungeon next to the player to move into.

Enter Command: p
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: None
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|
          |   |
          |   |
          |   |
|^|^|^|^|^|   |^|^|^|^|^|

2. cave
Boss: Present
Monster: Slime
Empty

|^|^|^|^|^|   |^|^|^|^|^|

Enter Command: Ctrl-D
Thanks for playing!
```

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 02_03 cs_dungeon
```

Stage 2.4 - Fight Monsters

Now we get to really get into our dungeon crawler game, let's fight some monsters! There are two ways to fight monsters, with physical attacks, or magical attacks. These attacks do not apply to boss-level monsters. So in **Stage 2.4**, you will need to implement these commands.

Physical attacks (`damage` field in `struct player`):

```
Enter Command: !
```

Magical attacks (`damage` multiplied by `magic_modifier` in `struct player`):

```
Enter Command: #
```

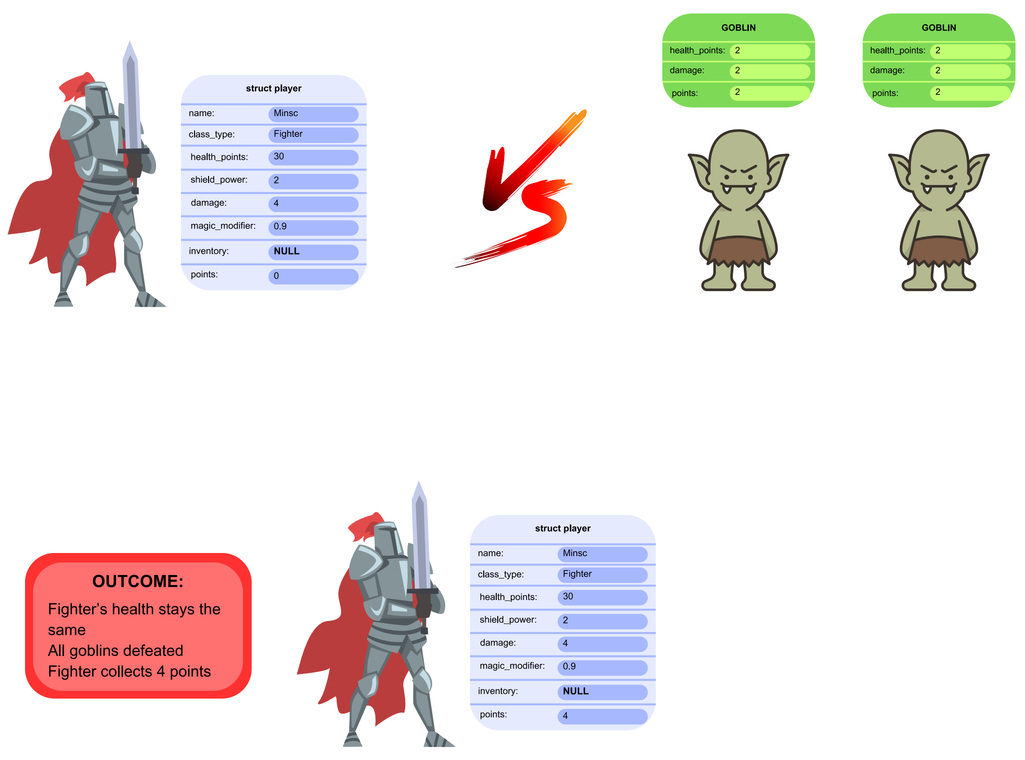
HINT:

There are a lot of fighting rules to understand for this stage. It may be helpful to refer to the diagrams, and to simulate some fights using the reference solution so you can get a better understanding. You can also read 'Rules: Short Version/Summary' below for a brief description without getting bogged down in the details. Try and approach this stage part by part, e.g. a single fight between the player and the monsters in the current dungeon first without shielding, then add shielding, then add collecting points, then add fights between turns, etc..

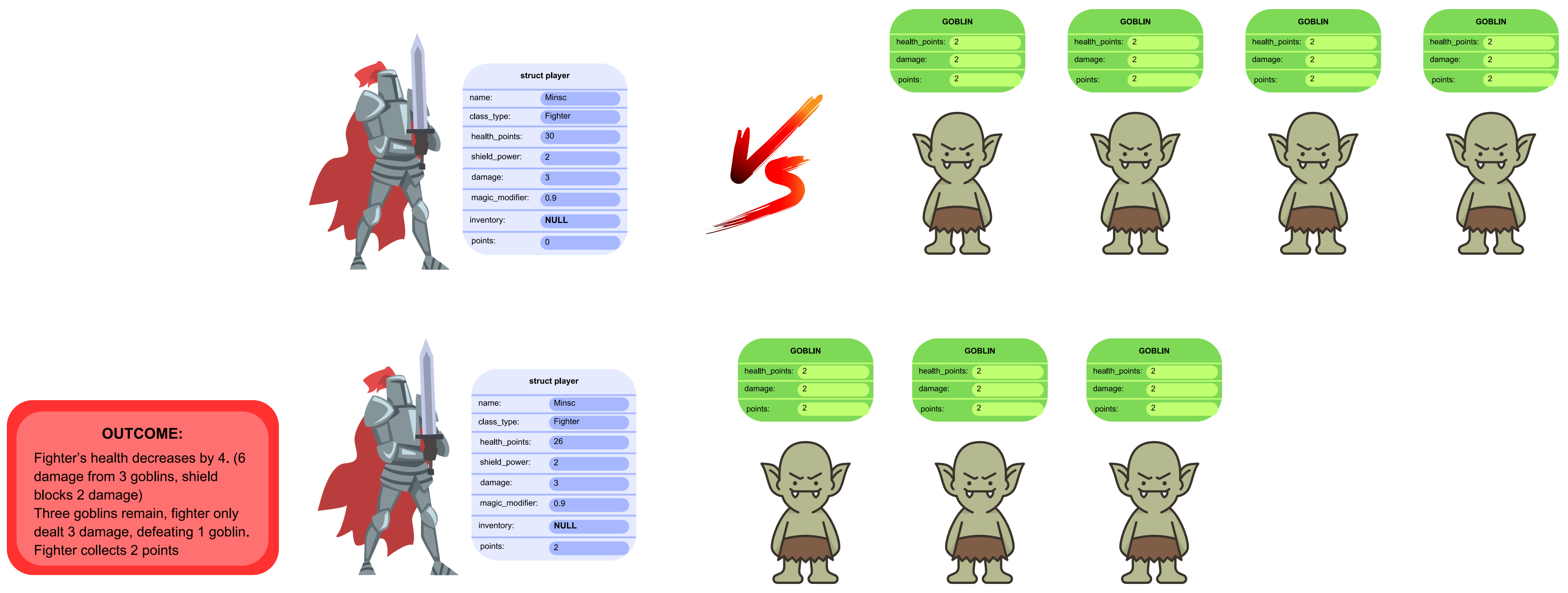
Rules: Short Version/Summary

- Monsters have health and attack damage based on their enum value, but the player's shield reduces some of the damage (based on shield power).
- The player can attack all monsters in the current dungeon with either a physical attack (`damage`) or a magical attack (`damage` multiplied by `magic modifier`).
- If any monsters survive the attack, they 'heal' and will be at full health for the next turn.
- Once monsters have been attacked, monsters in *that* dungeon will continue to attack the player each turn. Wolves, however, attack the player every turn, regardless of what happens.

Fighter with 4 damage fights 2 goblins:



Fighter with 3 damage fights 4 goblins:



OUTCOME:

Fighter's health decreases by 4. (6 damage from 3 goblins, shield blocks 2 damage)
Three goblins remain, fighter only dealt 3 damage, defeating 1 goblin.
Fighter collects 2 points

– Rules: Detailed

- Both a monster's health and damage is indicated by their enum value, e.g. slimes have 1 health point and do 1 damage.
- In a given dungeon, `monster` represents what type of monster is within, and `num_monsters` represents how many of that monster there are.
- When the player attacks, they can defeat as many monsters as they have damage to defeat. e.g. if the player had 4 damage, they could defeat 2 goblins (enum value of 2).
- If a player does not have enough damage to completely defeat a monster, the monster is not defeated. e.g. if the player had 3 damage, they could only defeat 1 goblin (enum value of 2).
- Any monster that was not defeated keeps their original health (so there is no monster health tracking between turns). e.g. if there were 3 goblins, and the player did 5 damage, there would still be one goblin left. Thematically, the monsters rest and heal between turns.
- The player should collect points after a battle totalling the number of monsters defeated multiplied by the monster's enum value.
- After being attacked for the first time, monsters will now fight the player at the end of each turn (when they are in the same dungeon), dealing damage equal to their enum value multiplied by the amount of monsters left. e.g. if there was 1 goblin left in a dungeon after fighting, at the end of each turn the goblin would deal 2 damage to the player (when they are in the same dungeon).
- If the player has any shielding, the damage from the monster is decreased by the amount of shielding the player has. e.g. if the player had 1 shielding, a goblin's damage would be 1.

There are two functions in `cs_dungeon.c` that you will have to implement for **Stage 2.4**, `fight` and `end_turn` . `fight` will handle the fight between the player and the monster in the current turn, and `end_turn` will handle any extra actions that will need to occur at the end of each turn, e.g. attacked monsters and wolves. This will be built upon further in later stages.

`fight` is called when a physical attack or magical attack command is inputted. `end_turn` is called by `main.c` at the end of each turn, after a command has been executed.

`fight` returns an `int` - this value indicates whether or not the fight was successful or not. Return `INVALID` if there are no monsters to attack, and `VALID` otherwise.

`end_turn` returns an `int` - this value indicates whether or not the game is over. Return `CONTINUE_GAME` for now, ending the game will be handled in **Stage 3.1**.

Examples

– Example 2.4.1: Defeat all monsters in dungeon

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
    Fighter
    Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a forest 3 3
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: Present
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 3 Skeletons
The boss is in this dungeon
    Health Points: 35
    Damage: 10
    Points: 20
    Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: !
A battle has raged!

Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 1 Skeletons
The boss is in this dungeon
    Health Points: 35
    Damage: 10
    Points: 20
    Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: !
A battle has raged!

Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 0 Skeletons
The boss is in this dungeon
```

```
    Health Points: 35
    Damage: 10
    Points: 20
    Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: Ctrl-D
Thanks for playing!
```

– Example 2.4.2: Defeat some monsters in dungeon, monsters attack back

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a forest 3 4
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: Present
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 4 Skeletons
The boss is in this dungeon
        Health Points: 35
        Damage: 10
        Points: 20
        Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: s
=======Player Stats=======
Minsc is currently in forest
Fighter
Health Points: 30
Shield Power: 2
Damage: 8
Magic Modifier: 0.9
Points Collected: 0
Minsc has the following items in their inventory:
No Items

Enter Command: !
A battle has raged!

Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 2 Skeletons
The boss is in this dungeon
        Health Points: 35
        Damage: 10
        Points: 20
        Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: s
=======Player Stats=======
Minsc is currently in forest
Fighter
Health Points: 22
Shield Power: 2
Damage: 8
Magic Modifier: 0.9
Points Collected: 6
Minsc has the following items in their inventory:
No Items

Enter Command: Ctrl-D
Thanks for playing!
```

Clarifications

- Shields do not decay.
- Wolves will fight the player without needing to be attacked first. This will need to be implemented in the `end_turn` function.
- Monsters attack **after** the player, so if the player defeats any monsters in their turn, those defeated monsters cannot attack the player (as they have been defeated).
- Monsters that have been attacked by the player will continue to attack when the player leaves and returns to their dungeon, their change of state persists.
- Only monsters that have been attacked will attack the player, monsters in other dungeons remain dormant and will not attack the player until the player attacks them specifically, except for the `WOLF` type monster.
- Monsters only do damage to the player in the `end_turn` function, which is called for you in `main.c` at the end of each turn.
- Any command entered counts as a turn (even player stats(`s`), display current dungeon details (`d`) and print map (`p`)), so monsters can attack on these turns (if they have already been attacked once by the player).
- Shielding applies to the total damage dealt by all monsters attacking, not each individual monster.

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 02_04 cs_dungeon
```

Stage 2.5 - Class Powers

Now we can unleash the special powers of the player's chosen class!

```
Enter Command: P
```

When the `P` command is entered, the player's class ability is activated. This can only be used once per game.

If the player is a **Fighter**, their damage is now permanently increased, 1.5 times.

If the player is a **Wizard**, they defeat all monsters in the current dungeon, except for the final boss.

If the special power has already been used this game, an error message should be printed and nothing should happen.

There is one function in `cs_dungeon.c` that you will have to implement for **Stage 2.5**, `class_power` .

This function returns an `int` - this value indicates whether or not using the class power was successful. Return `VALID` if the power was used, and `INVALID` if not.

Clarifications

- When a Wizard defeats all monsters in the current dungeon, they also collect all the points those monsters were worth.
- A power can only be used if it has not already been used this game (i.e. class powers can only be used once per game).

Examples

– Example 2.5.1: Fighter Power

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Fighter


----------Setup Phase----------


Enter Command: a forest 3 2
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: Present
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: s
=======Player Stats=======
Minsc is currently in forest
Fighter
Health Points: 30
Shield Power: 2
Damage: 8
Magic Modifier: 0.9
Points Collected: 0
Minsc has the following items in their inventory:
No Items

Enter Command: P
Minsc used their class power! This can no longer be used.

Enter Command: s
=======Player Stats=======
Minsc is currently in forest
Fighter
Health Points: 30
Shield Power: 2
Damage: 12
Magic Modifier: 0.9
Points Collected: 0
Minsc has the following items in their inventory:
No Items

Enter Command: Ctrl-D
Thanks for playing!
```

– Example 2.5.2: Wizard Power

```
$ dcc cs_dungeon.c main.c -o cs_dungeon
$ ./cs_dungeon
Welcome to the 1091 Dungeon!
This is a game where you get to create your own dungeon map, battle monsters and collect items!
Please enter the name of your map: Faerun
Please enter the amount of points required to win: 25
Please enter the player's name: Minsc
Player Class Options:
        Fighter
        Wizard
Please enter player's chosen class type: Wizard


----------Setup Phase----------


Enter Command: a forest 3 2
forest has been added as a dungeon to the map!

Enter Command: q
Please enter the required item to defeat the final boss: 0
The final boss has been added to Faerun!
Map of Faerun!
|^|^|^|^|^|   |^|^|^|^|^|

1. forest
Boss: Present
Monster: Skeleton
Minsc is here

|^|^|^|^|^|   |^|^|^|^|^|


--------Gameplay Phase--------


Enter Command: s
=======Player Stats=======
Minsc is currently in forest
Wizard
Health Points: 15
Shield Power: 0
Damage: 7
Magic Modifier: 1.5
Points Collected: 0
Minsc has the following items in their inventory:
No Items

Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 2 Skeletons
The boss is in this dungeon
        Health Points: 35
        Damage: 10
        Points: 20
        Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: P
Minsc used their class power! This can no longer be used.

Enter Command: d
======Dungeon Details======
Minsc is currently in forest
There are 0 Skeletons
The boss is in this dungeon
        Health Points: 35
        Damage: 10
        Points: 20
        Required Item: Physical Weapon
The dungeon forest has the following items:
No Items

Enter Command: s
======Dungeon Details======
Minsc is currently in forest
Wizard
Health Points: 15
Shield Power: 0
Damage: 7
Magic Modifier: 1.5
Points Collected: 6
Minsc has the following items in their inventory:
No Items

Enter Command: Ctrl-D
Thanks for playing!
```

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 02_05 cs_dungeon
```

Testing and Submission

Remember to do your own testing

Are you finished with this stage? If so, you should make sure to do the following:

- Run `1091 style` and clean up any issues a human may have reading your code. Don't forget -- **20%** of your mark in the assignment is based on style and readability!
- Autotest for this stage of the assignment by running the `autotest-stage` command as shown below.
- Remember -- *give early and give often*. Only your last submission counts, but why not be safe and submit right now?

```
$ 1091 style cs_dungeon.c
$ 1091 autotest-stage 02 cs_dungeon
$ give dp1091 ass2_cs_dungeon cs_dungeon.c
```

Assessment

Assignment Conditions

- **Joint work** is **not permitted** on this assignment.

  This is an individual assignment.

  The work you submit must be entirely your own work. Submission of any work even partly written by any other person is not permitted.

  The only exception being if you use small amounts (< 10 lines) of general purpose code (not specific to the assignment) obtained from a site such as Stack Overflow or other publicly available resources. You should attribute the source of this code clearly in an accompanying comment.

  Assignment submissions will be examined, both automatically and manually for work written by others.

  Do not request help from anyone other than the teaching staff of DPST1091/CPTG1391.

  Do not post your assignment code to the course forum - the teaching staff can view assignment code you have recently autotested or submitted with give.

Rationale: this assignment is an individual piece of work. It is designed to develop the skills needed to produce an entire working program. Using code written by or taken from other people will stop you learning these skills.

- The use of **code-synthesis tools**, such as **GitHub Copilot**, is **not permitted** on this assignment.

  The use of **Generative AI** to generate code solutions is not permitted on this assignment.

  Rationale: this assignment is intended to develop your understanding of basic concepts. Using synthesis tools will stop you learning these fundamental concepts.

- **Sharing, publishing, distributing** your assignment work is **not permitted**.

  Do not provide or show your assignment work to any other person, other than the teaching staff of DPST1091/CPTG1391. For example, do not share your work with friends.

  Do not publish your assignment code via the internet. For example, do not place your assignment in a public GitHub repository.

  Rationale: by publishing or sharing your work you are facilitating other students to use your work, which is not permitted. If they submit your work, you may become involved in an academic integrity investigation.

- **Sharing, publishing, distributing your assignment work after the completion of DPST1091/CPTG1391** is **not permitted**.

  For example, do not place your assignment in a public GitHub repository after DPST1091/CPTG1391 is over.

  Rationale:DPST1091/CPTG1391 sometimes reuses assignment themes, using similar concepts and content. If students in future terms can find your code and use it, which is not permitted, you may become involved in an academic integrity investigation.

Violation of the above conditions may result in an academic integrity investigation with possible penalties, up to and including a mark of 0 in DPST1091/CPTG1391 and exclusion from UNSW.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted - you may be penalised, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.

If you have not shared your assignment, you will not be penalised if your work is taken without your consent or knowledge.

For more information, read the UNSW Student Code , or contact the course account. The following penalties apply to your total mark for plagiarism:

0 for the assignment	Knowingly providing your work to anyone and it is subsequently submitted (by anyone).
0 for the assignment	Submitting any other person's work. This includes joint work.
0 FL for DPST1091	Paying another person to complete work. Submitting another person's work without their consent.

Submission of Work

You should submit intermediate versions of your assignment. Every time you autotest or submit, a copy will be saved as a backup. You can find those backups here , by logging in, and choosing the yellow button next to `ass2_cs_dungeon` .

Every time you work on the assignment and make some progress, you should copy your work to your CSE account and submit it using the `give` command below.

It is fine if intermediate versions do not compile or otherwise fail submission tests.

Only the final submitted version of your assignment will be marked.

You submit your work like this:

```
$ give dp1091 ass2_cs_dungeon cs_dungeon.c
```

Assessment Scheme

This assignment will contribute 25% to your final mark.

80% of the marks for this assignment will be based on the performance of the code you write in `cs_dungeon.c` .

20% of the marks for this assignment will come from manual marking of the readability of the C you have written. The manual marking will involve checking your code for clarity, and readability, which includes the use of functions and efficient use of loops and if statements.

Marks for your performance will be allocated roughly according to the below scheme.

100% for Performance	Completely Working Implementation, which exactly follows the specification (Stage 1, 2, 3 and 4).
85% for Performance	Completely working implementation of Stage 1, 2 and 3.
65% for Performance	Completely working implementation of Stage 1 and Stage 2.
35% for Performance	Completely working implementation of Stage 1.

Marks for your style will be allocated roughly according to the scheme below.

Style Marking Rubric

	0	1	2	3	4
Formatting (/5)					
Indentation (/2) - Should use a consistent indentation scheme.	Multiple instances throughout code of inconsistent/bad indentation	Code is mostly correctly indented	Code is consistently indented throughout the program		
Whitespace (/1) - Should use consistent whitespace (for example, 3 + 3 not 3+ 3)	Many whitespace errors	No whitespace errors			
Vertical Whitespace (/1) - Should use consistent whitespace (for example, vertical whitespace between sections of code)	Code has no consideration for use of vertical whitespace	Code consistently uses reasonable vertical whitespace			
Line Length (/1) - Lines should be max. 80 characters long	Many lines over 80 characters	No lines over 80 characters			
Documentation (/5)					
Comments (incl. header comment) (/3) - Comments have been used throughout the code above code sections and functions to explain their purpose. A header comment (with name, zID and a program description) has been included	No comments provided throughout code	Few comments provided throughout code	Comments are provided as needed, but some details or explanations may be missing causing the code to be difficult to follow	Comments have been used throughout the code above code sections and functions to explain their purpose. A header comment (with name, zID and a program description) has been included	
Function/variable/constant naming (/2) - Functions/variables/constants names all follow naming conventions in style guide and help in understanding the code	Functions/variables/constants names do not follow naming conventions in style guide and help in understanding the code	Functions/variables/constants names somewhat follow naming conventions in style guide and help in understanding the code	Functions/variables/constants names all follow naming conventions in style guide and help in understanding the code		
Organisation (/5)					

Function Usage (/4) - Code has been decomposed into appropriate functions separating functionalities	No functions are present, code is one main function	Some code has been moved to functions	Some code has been moved to sensible/thought out functions, and/or many functions exceed 50 lines (incl. main function)	Most code has been moved to sensible/thought out functions, and/or some functions exceed 50 lines (incl. main function)	All code h been meaningfu decompos into functions a maximu of 50 lines (incl. The main function)
Function Prototypes (/1) - Function Prototypes have been used to declare functions above main	Functions are used but have not been prototyped	All functions have a prototype above the main function or no functions are used			
Elegance (/5)					
Overdeep nesting (/2) - You should not have too many levels of nesting in your code (nesting which is 5 or more levels deep)	Many instances of overdeep nesting	<= 3 instances of overdeep nesting	No instances of overdeep nesting		
Code Repetition (/2) - Potential repetition of code has been dealt with via the use of functions or loops	Many instances of repeated code sections	<= 3 instances of repeated code sections	Potential repetition of code has been dealt with via the use of functions or loops		
Constant Usage (/1) - Any magic numbers are #defined	None of the constants used throughout program are #defined	All constants used are #defined and are used consistently in the code			
Illegal elements					
Illegal elements - Presence of any illegal elements indicated in the style guide	**CAP MARK AT 16/20**				

Note that the following penalties apply to your total mark for plagiarism:

0 for the assignment	Knowingly providing your work to anyone and it is subsequently submitted (by anyone).
0 for the assignment	Submitting any other person's work. This includes joint work.
0 FL for DPST1091	Paying another person to complete work. Submitting another person's work without their consent.

Allowed C Features

In this assignment, there are no restrictions on C Features, except for those in the style guide. If you choose to disregard this advice, you **must** still follow the style guide.

You also may be unable to get help from course staff if you use features not taught in DPST1091. Features that the Style Guide identifies as illegal will result in a penalty during marking. You can find the style marking rubric above. Please note that this assignment must be completed using only **Linked Lists** . Do not use arrays in this assignment. If you use arrays instead of lined lists you will receive a 0 for performance in this assignment.

Due Date

This assignment is due **Week 12 Friday 09:00am** (2025-04-11 09:00:00). For each day after that time, the maximum mark it can achieve will be reduced **by 5%** (off the ceiling). For example at:

- Less than 1 day (24 hours) past the due date, the maximum mark you can get is **95%**.
- Less than 2 days (48 hours) past the due date, the maximum mark you can get is **90%**.
- Less than 5 days (120 hours) past the due date, the maximum mark you can get is **75%**.

No submissions will be accepted at 5 days late, unless you have special provisions in place.

Change Log

Version 1.0 (2025-03-19 09:00)	• Assignment Released

DPST1091/CPTG1391 25T1: Programming Fundamentals!