INFO1910 Assignment 2

Due: 11:59PM Monday 31 May 2021 local Sydney time

This assignment is worth 20% of your final assessment

Platformer

In this assignment you will be using C to write a program for a puzzle game called 'platformer'. Platformer is a top-down two dimensional single player game. The goal is for the player to reach the exit. On the way they must contend with uneven terrain and moving platforms. Your program must both load a given level from a text file and manage the game as it is played.
You may organise your source code however you like, including dividing it into multiple files. You must provide a makefile with your submission, and it must compile into a single binary file called 'platformer'. When running the program, a level file must be provided as the first command line argument.

The Board

The board of Platformer is composed of tiles arranged in a rectangular shape. The player moves around on the board, trying to reach the exit. There are several different types of tiles, and the player's movement options are restricted by the tile types they move from and to. Here is a simple example board, as it would appear in the game:

```
P--/+++
a++++$+
X-A-/++
```

Tiles

Tiles are the basic components which make up the board. Tiles primarily define the physical topography of the level, and where a player is allowed to move. Tiles come in five main types: Low Ground, High Ground, Ramp, Platform, and Wall.
Low ground: −
High ground: +

Walls appear as spaces in the game and as asterisks in level files: `*`
Ramps come in four varieties.
South to North (low ground at the south, high ground at the north): `#`
North to South (low ground at the north, high ground at the south): `=`
East to West (low ground to the east, high ground to the west): `\`
West to East (low ground to the west, high ground to the east): `/`
Platforms are the most complicated type of tile. Platforms are organised in groups which are referred to by letters of the alphabet. There may be up to eight groups of platforms in a level; the letters `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H`. Platforms have two states, 'active' and 'inactive'. While inactive, a platform appears and behaves like low ground. When active, a platform is represented by the capital letter of its assigned group, and behaves like high ground. Thus, the above example board would appear like this while the 'A' platforms are active:

```
P--/+++
a++++$+
X-A-/++
```

But if the 'A' platforms are switched to inactive mode, it would appear like this:

```
P--/+++
a++++$+
X---/++
```

Platform state is changed when the player uses switches (see below).

Special Objects

Tiles deal with the physical layout of the level, but special objects deal with the interactive elements of the level. There are four types of special object in the game: the player, the exit, switches, and keys. Special objects may be placed on most tiles types, but must not be placed on walls or platforms, and at the beginning of the game no two special objects may be positioned on the same tile. (The player is the only special object which can move around).
The player can move around, pick up keys, toggle switches, and wins the game by stepping on the exit object: `P`
The exit: `X`
Switches are used to change the state of the relevant group of platforms. Switches are represented in the level by the lower case version of the letter group for the platforms they affect: `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`.
Keys are necessary in order to use switches. There is a different key for every switch in the level, but all keys appear the same during gameplay until the player picks them up: `$`.
The player picks up keys by moving onto the tile the key is on. The player must move onto the same

tile as a switch before toggling it. When the player moves onto the same tile as the exit, the game is won.

Player Movement

In general, a player may move north, south, east, and west, but not diagonally. However, player movement is restricted by the tiles of the level.
A player may only move onto a tile if that tile is behaving as the same level of ground as the player's current tile. For instance, a player may move from low ground onto low ground or onto an inactive platform, but not onto an active platform (which is behaving as high ground). Similarly a player may move from high ground onto high ground or an active platform, but not onto an inactive platform.
Every ramp has a front and back end. Front is the end of the ramp which behaves as low ground, back is the end of the ramp which behaves as high ground.
North to South ramp: Front = North, Back = South
South to North ramp: Front = South, Back = North
West to East ramp: Front = West, Back = East
East to West ramp: Front = East, Back = West
The player may only move onto the front of a ramp when they will be moving off a tile which behaves as low ground, and may only move onto the back of a ramp when moving off a tile which behaves as high ground. Equivalent rules apply when moving off the front and back of a ramp tile. Note that you cannot move off the front of a ramp onto the front of another ramp, or equivalently with the back of ramps, but this is enforced by the loading rules.
You may only move off the side of a ramp (the two ramp ends which are not the front or back) when you are moving onto another ramp of exactly the same type. Otherwise, ramps cannot be entered or left through the sides.
Walls are inaccessible by any means whatsoever. There is no way to move on to them and no special objects may be placed on them.
The player cannot move off the edge of the board. Boards need not be surrounded by walls.
If the player attempts to move in a direction which they are not allowed to, nothing happens. The player does not move and no error message is printed.
Special objects have no affect on a player's ability to move.

Whenever a player moves, if the player lands on a key, they collect that key and carry it with them for the rest of the game. When a key is picked up, a message is printed:
`You picked up the key for switch a!`
with the letter for the relevant switch substituted instead of 'a'.
When the player moves on to a switch, they may attempt to toggle the switch. If the player holds the relevant key, the switch will toggle and all platforms of the relevant letter will either activate or deactivate, switching from the state they were in previously to the other. If the player did not have the key for that switch, the message
`You do not have the key for that switch!` will print.

Winning and Losing

When the player moves onto the tile containing the exit, the game immediately ends with the message `Congratulations, you win!`
There is no mechanic for the player to lose the game, but the player can quit at any time. See 'Game Commands' below.

Loading

The program loads levels from files. When the program starts up, it expects to receive a single command line argument giving the name of a level file to load. Here is an example of a level file:

```
**********
*---/+++**
*+++++++**
*--A-/++**
**********

P 1 1
X 1 3
a 1 2 6 2 1
```

Remember that walls are represented as asterisks (*) in level files, but are represented by spaces in the game.
The first portion of the level file defines the topography or tiles which make up the level. After the tile section, there is a single blank line, followed by the lines defining the special objects. The special objects may be listed in any order, but there may not be blank lines in between them.

Loading the Tiles

The program should process the tile section from left to right of elements in a row, and from top to bottom through the rows. The only valid tiles for this section are low ground, high ground, * to represent walls, the four types of ramps, and the eight types of platforms. Note that the presence of a platform in the level file does not imply that the platform will necessarily be active when the game begins.
* is used to represent walls in level files to make it easier to write level files where all rows are the same length, since this is a requirement for a valid level file.

Loading the Special Objects

The special object that a line is giving information about is specified by the first character of the line. The player and exit special objects are referred to by `P` and `X` respectively, and require two integers in their definition. These integers are the `x` and `y` coordinates on the board where the object should be placed.
Switches are identified by the lower case letter of their associated platform group, and require five integers in their definition. The first two integers are the `x` and `y` coordinates of the switch. The next two integers are the `x` and `y` coordinates of the associated key. The final fifth integer is either 0 or 1, and represents whether platforms of that group should begin the game active (1) or inactive (0).

Coordinates for all special objects begin at (x, y) = (0, 0), and thus go up to a maximum of (59, 19) as the board has a maximum size of 20 rows and 60 columns. Coordinates begin in the top left corner of the board and increase to the right and downwards.

Loading Errors

If any of these errors occurs, the program should print the relevant error message and exit immediately.

If you attempt to run the program with no file provided as a command line argument, it should print
```
Usage: ./platformer <level file>.
```
If you provide as a command line argument the name of a level file which does not exist:
```
./platformer no_file.txt
```
the program should print `File no_file.txt does not exist.`

If at any point in the tile section of the file a character is found which does not represent a valid tile,
```
Found invalid tile at position 0, 7 in the level.
```
will be printed, providing the x and y coordinates of the invalid object on the board. Coordinates start from 0 at the top left, and increase to the right and downwards.
The length of the first row defines the number of columns the game should have. Every row in the tile section of the level file must be the same length as the first row. If a row of incorrect length is found, the message
```
Row 3 of the level had incorrect length 15; length should have been 12.
```
will print, with appropriate lengths and row index substituted.
No board may ever have a row longer than 60 tiles. If a row is found to be longer than 60 tiles, the message
```
Row 5 is longer than maximum row length 60.
```
will print, substituting the appropriate row index.
No board may ever have a column longer than 20 tiles. If more than 20 rows appear in the level, the message
```
Level has more rows than maximum row number 20.
```
will print.
Row and columns indexes on the board always start from 0, and thus go up to a maximum of 19 and

59 respectively.

If the special object section of the level file is missing entirely, the program should print

```
The file ended before any special objects had been defined.
```

If one of the special object lines begins with a character that is not a valid special object, the program should print, giving the offending character, that

```
Level file contained invalid object i.
```

If a special object line does not have enough integers to describe the object being defined (2 for the player or the exit, 5 for a switch), the messages

```
Could not find 2 integers for player.
Could not find 2 integers for exit.
Could not find 5 integers for switch a.
```

should print, with switch letters substituted as appropriate.
Special objects may be placed on ramps, high ground, and low ground, but not on walls or platforms. If a special object is placed on a wall or platform, the following messages should print, with key and switch letters changed as appropriate:

```
Player P was placed on a platform.
Exit X was placed on a platform.
Switch f was placed on a platform.
Key g was placed on a platform.
A special object was placed on a wall at 0, 0.
```

The player and exit objects must be defined, for every group of platforms there must be switch, and for every switch there must be at least one platform in the level that it will affect. If any of these conditions are not met, the program will accordingly print one of the following:

```
No player is defined in the level file.
No exit is defined in the level file.
No platform exists for switch h.
No switch exists for platform A.
```

As usual, the switch and platform letters must be substituted as necessary.

Ramps must be placed with the correct tile types at their front and back ends. The front of a ramp may be adjacent to low ground or a platform, and the back of a ramp may be adjacent to high ground or a platform, but nothing else. The sides of a platform may be adjacent to any kind of tile. In particular, ramps may not lead on to other ramps. If a ramp is placed with the incorrect kind of tile at one of its ends, the program should print a message like

```
Ramp at 4, 1 does not start and end on the correct levels.
```

using the appropriate coordinates.
Ramps also may not lead into the edge of the level map. If the front or back of any ramp is adjacent to the edge of the board, the message

```
Ramp at 0, 2 leads off the board.
```

should print with appropriately changed coordinates.

Special objects may not overlap. If any two special objects are placed on the same tile when the level is loaded, the program should print

```
Two special objects are placed at the same coordinates 1, 3.
```

with the appropriate coordinates. Note that the player is the only special object that can move, and that the player may move onto tiles containing switches, keys, and the exit at different times throughout the game; it is only at loading that no overlap is permitted.

You will not be tested on the precendence of multiple different types of errors. It is enough that no invalid file will load successfully, and any file with only one type of error will give the correct error message.

Game Commands

Once loading has completed successfully, the program should print the game board in its initial state. Every time the user enters a command (whether or not it succeeds), the program will print any applicable message, then print a single blank line and print out the board again.
The user gives commands to the game by typing a letter and pressing enter. All valid commands in the game are single letters. All commands are case-sensitive, and full words, including words containing the letters for valid commands, will not be accepted, although whitespace around a command will be ignored.
If the game does not recognise the command the user has given, it will print
`Command not recognised.`
print a blank line, and reprint the board, unchanged.

The command `h` prints the help message

```
w: Move up one space.
s: Move down one space.
a: Move left one space.
d: Move right one space.
x: Toggle the state of the switch you are standing on.
k: See which keys you have collected.
e: See which platforms are on and which platforms are off.
h: Print this help message.
q: Quit the game.
```

The command `q` immediately quits the game and ends the program; nothing further is printed.

The commands `e` and `k` display which platforms are on and which are off, and which keys the player has collected and which they have not, respectively. Only the platforms and keys which actually exist in the current level are displayed. 1 indicates a platform group is active or that the player holds a key; 0 indicates the platform group is inactive or that the player does not hold a key.
`e` would print something like

```
A: 1
B: 0
```

```
C: 1
D: 0
E: 1
F: 0
G: 1
H: 0
```

while `k` would print something like

```
a: 1
b: 1
c: 1
d: 1
e: 0
f: 0
g: 0
h: 0
```

The command `x` is used to toggle a switch that the player is currently standing on. If the player is not standing on a switch, the game will print
`You are not standing on a switch!`
If the player is standing on a switch but does not hold the relevant key, the program will print
`You do not have the key for that switch!`
If the player is standing on a switch and has the right key then no message will be printed, but all platforms of the relevant group will immediately switch to the opposite state.

The player is moved up, down, right, and left (or north, south, east, and west) using the keys `w`, `s`, `d`, and `a` respectively.
If the player attempts to make a move which is not legal (see the 'Player Movement' section), no message will be printed, but the player will not move.
If, in making the move, the player lands on a tile containing a key, the program will print
`You picked up the key for switch b!`
substituting in the correct switch letter.
If, in making the move, the player lands on the same tile as the exit, the program will print
`Congratulations, you win!`
and the game and program will end immediately.

Compilation and Execution

MAKE SURE TO INCLUDE A MAKEFILE WITH YOUR SUBMISSION. Without a makefile your code cannot compile, and if your code does not compile then you cannot receive any marks for automated testcases. Be sure to test your makefile on Ed; not all systems compile C programs the same way. If your program compiles on your own machine but not on Ed, you will still receive 0 marks for

automated testcases!
Your compiled code should produce one binary called 'platformer'. As explained above, this binary will take in one command line argument specifying the file from which to load a level. You may organise your source code however you want, provided it all compiles to one binary. Please don't upload any unused source code or unnecessarily split your code up across a large number of different files though; these things will make it harder to mark your work.

Your program will be run like this:

```
./platformer example_level.txt
```

Where `example_level.txt` is substituted with the name of whatever file contains the level currently being tested.

Examples

Simple Level

```
 P--/+++
 a++++$+
 X-A-/++

d


 -P-/+++
 a++++$+
 X-A-/++

d


 --P/+++
 a++++$+
 X-A-/++

d


 ---P+++
 a++++$+
 X-A-/++

d
```

```
 ---/P++
 a++++$+
 X-A-/++

d


 ---/+P+
 a++++$+
 X-A-/++

s
You picked up the key for switch a!


 ---/+++
 a++++P+
 X-A-/++

a


 ---/+++
 a+++P++
 X-A-/++

a


 ---/+++
 a++P+++
 X-A-/++

a


 ---/+++
 a+P++++
 X-A-/++

a


 ---/+++
 aP+++++
 X-A-/++

a


 ---/+++
 P++++++
 X-A-/++

x


 ---/+++
 P++++++
 X---/++

d


 ---/+++
 aP+++++
 X---/++

d


 ---/+++
 a+P++++
 X---/++

d


 ---/+++
 a++P+++
 X---/++

d


 ---/+++
 a+++P++
 X---/++

d
```

```
 ---/+++
 a++++P+
 X---/++

s


 ---/+++
 a++++++
 X---/P+

a


 ---/+++
 a++++++
 X---P++

a


 ---/+++
 a++++++
 X--P/++

a


 ---/+++
 a++++++
 X-P-/++

a


 ---/+++
 a++++++
 XP--/++

a
Congratulations, you win!
```

Giving Up

```
     --P--  a
  X  -----------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
s

     -----  a
  X  --P--------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  ---P-------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  ----P------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -----P-----
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  ------P----
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -------P---
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
w

     -----  P
  X  -----------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
x
You do not have the key for that switch!

     -----  P
  X  -----------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
h
w: Move up one space.
s: Move down one space.
a: Move left one space.
d: Move right one space.
x: Toggle the state of the switch you are standing on.
k: See which keys you have collected.
e: See which platforms are on and which platforms are off.
h: Print this help message.
q: Quit the game.

     -----  P
  X  -----------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
s

     -----  a
  X  -------P---
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
```

```
a

     -----  a
  X  ------P----
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
k
a: 0
b: 0

     -----  a
  X  ------P----
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -------P---
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
w

     -----  P
  X  -----------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
x
You do not have the key for that switch!

     -----  P
  X  -----------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
s

     -----  a
  X  -------P---
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
k
a: 0
b: 0

     -----  a
  X  -------P---
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
a

     -----  a
  X  ------P----
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
a

     -----  a
  X  -----P-----
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
a

     -----  a
  X  ----P------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
a

     -----  a
  X  ---P-------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
a
```

```
     -----  a
  X  --P--------
  +  $----   = -
  +++AAAAA++++ B
----------------
b--------------$
s

     -----  a
  X  -----------
  +  $-P--   = -
  +++AAAAA++++ B
----------------
b--------------$
a

     -----  a
  X  -----------
  +  $P---   = -
  +++AAAAA++++ B
----------------
b--------------$
a
You picked up the key for switch a!

     -----  a
  X  -----------
  +  P----   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -----------
  +  -P---   = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -----------
  +  --P--   = -
  +++AAAAA++++ B
----------------
```

```
b--------------$
d

     -----  a
  X  -----------
  +  ---P-    = -
  +++AAAAA++++ B
----------------
b--------------$
w

     -----  a
  X  ---P-------
  +  -----    = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  ----P------
  +  -----    = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -----P-----
  +  -----    = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  ------P----
  +  -----    = -
  +++AAAAA++++ B
----------------
b--------------$
d

     -----  a
  X  -------P---
  +  -----    = -
  +++AAAAA++++ B
```

```
----------------
b--------------$
d

     -----  a
  X  --------P--
  +  -----   = -
  +++AAAAA++++ B
----------------
b--------------$
a

     -----  a
  X  -------P---
  +  -----   = -
  +++AAAAA++++ B
----------------
b--------------$
w

     -----  P
  X  -----------
  +  -----   = -
  +++AAAAA++++ B
----------------
b--------------$
x

     -----  P
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b--------------$
a

     -----  P
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b--------------$
s

     -----  a
  X  -------P---
  +  -----   = -
  +++-----++++ B
----------------
b--------------$
a

     -----  a
  X  ------P----
  +  -----   = -
  +++-----++++ B
----------------
b--------------$
a

     -----  a
  X  -----P-----
  +  -----   = -
  +++-----++++ B
----------------
b--------------$
a

     -----  a
  X  ----P------
  +  -----   = -
  +++-----++++ B
----------------
b--------------$
s

     -----  a
  X  -----------
  +  ----P   = -
  +++-----++++ B
----------------
b--------------$
s

     -----  a
  X  -----------
  +  -----   = -
  +++----P++++ B
----------------
b--------------$
s

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
---------P------
b--------------$
s

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b--------P-----$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b-------P------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b------P-------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b-----P--------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b----P---------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b---P----------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b--P-----------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
b-P------------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
bP-------------$
a

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
P--------------$
x
You do not have the key for that switch!

     -----  a
  X  -----------
  +  -----   = -
  +++-----++++ B
----------------
P--------------$
k
```

```
a: 1
b: 0

     -----  a
  X  -----------
  +  -----    = -
  +++-----++++ B
----------------
P--------------$
d

     -----  a
  X  -----------
  +  -----    = -
  +++-----++++ B
----------------
bP-------------$
e
A: 0
B: 1

     -----  a
  X  -----------
  +  -----    = -
  +++-----++++ B
----------------
bP-------------$
q
```

Marking Criteria

The following is the marking breakdown, each point contributes a portion to the total 20% of the assignment.

Marks are allocated on the basis of:

- Correctness - 8 - Visible automatic test cases
- Correctness - 4 - Hidden automatic test cases
- Code structure - 2 - Use of functions
- Code style - 2 - Commenting / readability
- Code style - 2 - Error handling of C library functions
- Use of the preprocessor - 1 - Macros, custom header files

- Free resources when finished with them - 1

Warning: Any attempts to deceive or disrupt the marking system will result in an immediate zero for the entire assignment. Negative marks can be assigned if you do not follow the assignment description or if your code is unnecessarily or deliberately obfuscated.

Hints and Restrictions

Your program must be entirely written in C. There will be no python in this assignment. If you attempt to solve the assignment in python, none of the automatic testcases will work and you may receive 0 for the assignment.
You may not use any dynamic memory tools of any kind; this includes alloc functions, brk, sbrk, and mmap. You are not allowed to use any temporary files or variable length arrays. The program should not have any memory leaks.
Other than dynamic memory and similar tools, you may use any C standard library functions. You may not use external libraries.
Notice that there are several pieces of information that are not always visible on the board, such as the kind of tile underneath keys, switches, the player, and the exit. Your program will need to track this information in the background to ensure correct behaviour of the game. It is not enough to only track the information which is displayed on the board.

Academic declaration

By submitting this assignment you declare the following: