a) Through this project, we hope to construct a programming language that will allow an inexperienced coder to create a grid structured game. We draw inspiration from Dungeon Crawl games, but the exact structure of the game is be determined by the user. With this, we name our programming language Crall.

   This programming language will offer a more intuitive approach to programming this type of game. While these games can be coded using other programming languages, the process of creating such a game could require an extensive amount of coding experience. We hope to design a programming language that will simplify this process significantly, and offer an interactive introduction to programming.

b) Since our language is focusing on the creation of a game board that will consist of different types objects that exist in a 2D space, the aesthetic of our language will be most similar to that of object oriented programming. Object oriented is a particularly good fit for our language due to the fact that it will be more geared toward beginning programmers, for who a more object oriented approach might make more intuitive sense. Each new thing that you want to add contains the code for its functionality within its own structure block which we believe will help keep things simple conceptually.

c)
```
Example 1:
//This example is a simple 5x5 map with one
//pathway down the center from the start to end.
//This example features a vertical wall

dimension: 5x5 //Set the map as a 5x5 2D array
start: (2,0) //Set start position to (2,0)
goal: (2,3) //Set goal for map to (2,3)
wall: (1,0),(3,0) //Create wall from (1,0) to (3,0)

Example 2:
//This example is a 5x5 map that has no walls
//but a simple trap that ends the game if the
//player walks over it
dimension: 5x5
start: (2,0)
goal: (2,3)
trap: (2,2),20
//Set a trap at point [2,2] that does 20 damage

Example 3:
//This example sets walls that take up a
//rectangular space by giving
//two points that are not in a line
//This map features a horizontal wall
dimension: 20x20
start: (0,0)
```

```
goal: (0,5)
wall: (0,1),(0,4)
```

All of the above examples can be run by passing in a file, such as:

```
dotnet run example_1.crall
```

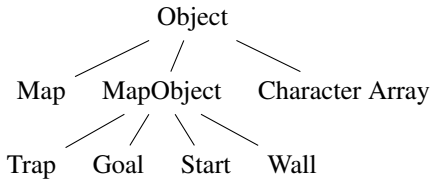
d) This programming language is designed for the more inexperienced user, so the user does not need to understand much. We aim to design this programming language in a simplified manner. The user will only need to understand function calls and two-dimensional array structure. Although the objects are actually stored in a map, we relieve the programmer of the burden of having to know how maps work, as we convert the map into an array for them. Through this programming language, we hope to offer users an entertaining medium for learning elementary computer programming.

e) The syntax for our language is going to be comprised almost entirely of function calls. In order to create a game, the user must first specify the dimensions of the array, and any edit to the map will simply require the user to use functions to set the starting point, goal, walls, and other objects in their desired locations. From here the user will type basic commands, such as "up" or "right" to control where their character moves, allowing the user to play the game.

f) Our programming language will feature the following primitives: integers, and characters. The user will specify dimensions and locations of objects by providing a coordinate. These coordinates will be in the form (int, int). All objects in the map are represented by ASCII characters.

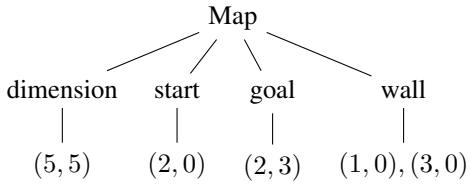
The language features commands when the user begins playing the constructed games. These commands will be something like move up/down/left/right to move the character around the Map. After creating the game, the user can initialize game play, and use these commands to play the game.

The program is represented using the following class hierarchy.

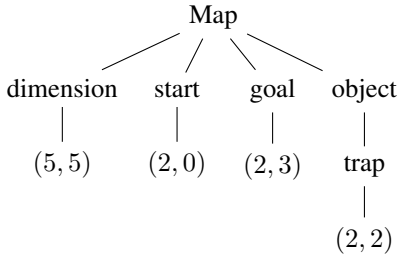


Enemies and walls will be represented using characters, and the Map will be composed of coordinates as an array.

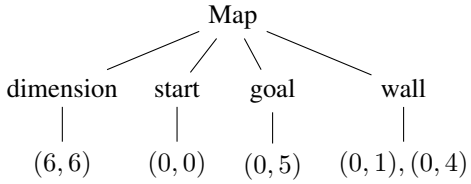
The abstract syntax tree for example 1 can be shown as



The AST for example 2 can be shown as



Lastly, the AST for example 3 can be shown as



A program in our language does take in an input, which is simply the user directing where they wish to move their character through the current map. When a program is evaluated, it will display a game board as designed by the programmer, and allow for the player to traverse it. The program will simply display the map and allow for the player to traverse it with their character, updating it using the code defined in the various structures as the user interacts with them. In this case, since the walls form a single pathway, the player will be limited in their movement depending on the objects set up in the map.

g)

```
<expr>          ::= <SeqOP>
                  |  <SetDim>
                  |  <SetGoal>
                  |  <SetStart>
                  |  <SetWall>
                  |  <SetLongWall>
                  |  <number>
                  |  <SetTrap>
<number>        ::= 0 | 1 | 2 ... 97 | 98 | 99
<SeqOP>         ::= <expr> <expr>
```

Syntax	Abstract Syntax	Type	Meaning
dimension: nxm	Two numbers of int	int*int	This is used to set the dimensions that the map will have
start: (n,m)	Two numbers of int	int*int	This is used to set the coordinates of the starting position to (n,m)
goal: (n,m)	Two numbers of int	int*int	This is used to set the coordinates of the end goal to (n,m)
wall: (n,m)	Two numbers of int	int*int	This is used to create a wall at the coordinate (n,m)
wall: (n,m),(x,y)	Four numbers of int	(int*int)*(int*int)	This is used to create a wall that extends from coordinate (n,m) to coordinate (x,y)
trap: (n,m),d	Three numbers of int	int*int*int	This is used to create a trap at coordinate (n,m) that does d damage

```
<SetDim>          ::= dimensions: <number>x<number>
<SetStart>‘       ::= start: (<number>,<number>)
<SetGoal>         ::= goal: (<number>,<number>)
<SetWall>         ::= wall: (<number>,<number>)
<SetLongWall>     ::= wall: (<number,<number>),(<number>,<number>)
<SetTrap>         ::= trap: (<number,<number),<number>
```

h)

i) The remainder of our work will focus on the game play portion of the code. Technically, the language is complete, for users can simply enter their .crall file and materialize their desired map. Playing the map will be a somewhat separate functionality of our program, since the user will not be programming in .crall, rather, the user will simply input up/down/left/right to move the character in the map and play the game.