**Jacobs – 217130232**

**Game: Ascii-Heroes**

My aim for the game was to make a game using only what we had been taught. I was mostly successful and ended up using only a few sections of more advanced coding. Since we have only been taught how to use the console, my entire game is played using the console. It is called Ascii-Heroes.

The entire game makes use of ascii-characters for the menu designs, enemies and characters. The game is a turn-based “action” game where the player controls the @ symbol and can perform one of eight actions in their turn before all the enemies take their own turn. The player must clear the level of all enemies before proceeding to the next level.

The menus are stored in text files and are written to the console each time they are loaded. At certain lines, a special line will be drawn and this is used to control the menu so that the player can choose menu options. This brings us to the first piece of coding which we hadn’t been taught but it is very simple. It is basically just a way of reading a specific key that is pressed and is used for all controls in the game. The command is Console.ReadKey(). It functions much like Console.ReadLine() except it stores a value for the key that was pressed. In my game, the arrow keys and enter were used for controlling menus and also the player.

The player can move or shoot in one of the four cardinal directions as part of their turn. The game is made up of a grid of chars which are stored in a 2D array of chars where the row and column in the array represents the same row and column in the game. Each turn the grid is updated based on the player’s actions and the enemies’ actions and then rewritten to the console. The enemies are all objects stored in an enemy list and they are differentiated by a string value called type which controls how the enemies move and attack. The one design choice I would like to bring up is the use of sorting for enemy movement. Before moving any enemies, I sort the enemy list by distance to the player. The reason behind this is so that enemies that are closer to the player move first. When enemies move, I check to see if the place they are moving to is empty. Making the closer enemies move first means that they will open up spaces for the enemies further away to move into, otherwise enemies near the back would get stuck as they wouldn’t be able to find any free spaces to move into.

The second section of more advanced coding that I used was used to control levels. The entire game is all within a single recursive function called RunGame. The function will run a level which is taken as a parameter into the function and then it will return the player’s score when they win or die. Inside the function, the level is loaded from a text file or is randomly generated using another function I wrote. The player will then play through the level which is all inside a while loop and the while loop will exit if the player kills all enemies in the level or dies. If the player dies, the function will return their score and the game will return to the menu. However, if the player killed all the enemies, then the RunGame function will be returned with a value of (Level + 1) as the level parameter. The player will then recursively play through the levels until they die or they reach a specific level and then the game will end because they won.

A few other things I added are:

* A Bestiary List: This allows the player to view all enemies they have killed as well as how each enemy works.
* Sound: The game has background music which is the last thing I added which we weren’t actually taught.
* Upgrades: As the player kills enemies, they will earn experience points which they can use to upgrade things like their health and damage.
* Score Attack Mode: This mode will generate a random level of varying difficulty and instead of earning experience points, the player will get a score which could be added to a high scores list which can be viewed from the menu.
* Saving: All the upgrades and experience points the player has earned is stored in a text file which allows the player to save their progress upon exiting. I also added an option to reset their save so that they can start again.