BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Group Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Chance**	**Nicander Alworth**	**2101693113**
	2.	**Herlie**	**Yoksan**	**2101684525**
	3.	**Chance**	**Sherlyn Angelia**	**2101693100**

Course Code	**: COMP6048**	**Course Name**	**: Introduction to Programming**
Class	**: L2AC**	**Name of Lecturer(s)**	: 1. Tri Asih Budiono 2. Raymondus Raymond Kosala
Major	**: CS**		
Title of Assignment (if any)	: Pacman		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 30-05-2018**	**Submission Date**	**: 29-05-2018**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:	(Name of Student)
1. Nicander Alworth Chance	
2. Yoksan Herlie	
3. Sherlyn Angelia Chance	

“PACMAN”

I. Description

A. The objective of this project is to create and program a pacman game using C++. Here, we implemented algorithms such as the Breadth First Search to implement this game. Aside from the algorithm, we also implemented the things we studied in class such as class and inheritance, struct, functions including STLs such as vector and arrays. This game is a modified version of Pacman. However, we use different approach and idea behind this modified pacman game.

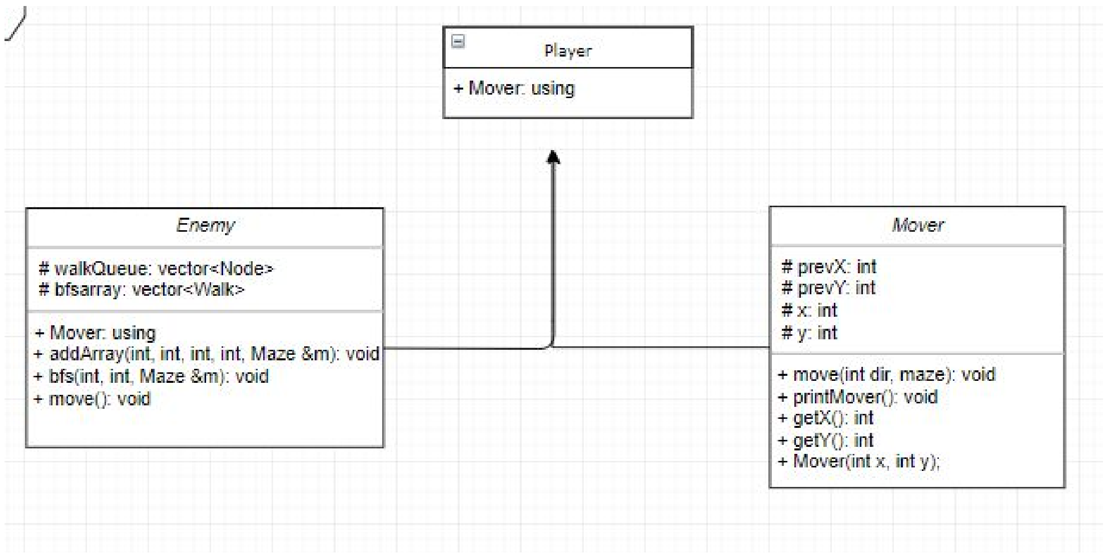
II. Our choice of Data Structures

A. The data structure that we used for this project is a graph. Because this game is a game based on the map where the player can traverse the map and the enemy will chase after the player. So, the graph represents the game map as a maze in a 2d array. With this data structure, we can implement the Breadth-First-Search algorithm for the enemy to chase after the player.

III. Theoretical Analysis of Data Structure

A. Graph is a set of data structure where each set of data (nodes) are connected by what we call vertices. We then use this data structure to implement the Breadth-First-Search algorithm for the enemy in the game. The theoretical analysis is that the data that is stored in each node are visited level by level , just like a binary tree. Nodes are visited from left to right at each level. Therefore, sorted result will be produced.

IV. UML Diagram



V. Explanation of each source code

A. Main.cpp
   1. This is the main file of the project which will be executed.
   2. Holds function such as detectKey(Player &player, Maze m), findRandomEnemyPos (Maze m), and the main().
   3. Int main():
      a) The main function uses a while loop to run the program.
      b) All the functions created in other files and classes are executed here such as the generateMaze() and printMaze().
      c) Under the while loop we have the game's elements running, such as the player to be printed, enemy to be moved, and the effect that will happen when the enemy catches the player. These are put under the while loop to allow the game to operate until a certain condition is reached. Also, the use of sleep is to pause the program for a certain amount of time, so that the user can play the game.

B. Mover.h
   1. Declaration of class member class attributes, methods, and constructors.
   2. Public:
      a) void move(int, Maze)
      b) void printMover()
      c) int getX()
      d) int getY()
      e) Mover(int x, int y)
   3. Protected:
      a) int prevX
      b) int prevY
      c) int X
      d) int Y

C. Mover.cpp
   1. Mover::Mover(int x, int y): constructor of the class
   2. int getX(): function to get the x coordinate of the mover
   3. int getY(): function to get the y coordinate of the mover
   4. void move(int dir, Maze m): function to move the mover position to the direction from the parameter
   5. void printMover(): print the mover in the maze

D. Enemy.h
   1. Consist of a struct Walk and a class called Enemy which inherits the Mover class.
   2. Declaration of class member class attributes, methods, and constructors.

3. Public:
    a) using Mover::Mover;
    b) void addArray(int, int, int, int, Maze &m);
    c) void bfs(int, int, Maze &m);
    d) void move();
4. Protected:
    a) vector<Node> walkQueue;
    b) vector<Walk> bfsArray;

E. Enemy.cpp
1. This is the file that accounts for all the data for the enemy's algorithm and movement
2. Holds function such as move(), bfs(), and addArray()
3. Void addArray(int x, int y, int wc, int back, Maze &): function to add struct walk object to the bfs Array if the coordinate is not visited/walkable.
4. void bfs(int playerX, int playerY, Maze &m) : function to implement the search algorithm by using the addArray() function to add elements to the array when certain coordinates are met

F. Player.h
1. Inherits the Mover class
2. Declaration of class member attributes, methods, and constructors.
3. Public:
    a) using Mover::Mover

G. Maze.h
1. Declaration of the Maze class attributes, methods and constructors.
2. Private:
    a) int grid[20][20];
    b) int size;
    c) vector<Node> walls;
    d) void addWalls(int x, int y);
    e) void initGrid();
    f) bool inMaze(int x, int y);
3. Public:
    a) void printMaze();
    b) void generateMaze();
    c) bool isWalkable(int, int);
    d) bool isWalkablePlayer(int, int);
    e) void removeVisited();
    f) int getSize();
    g) void setVisited(int, int);

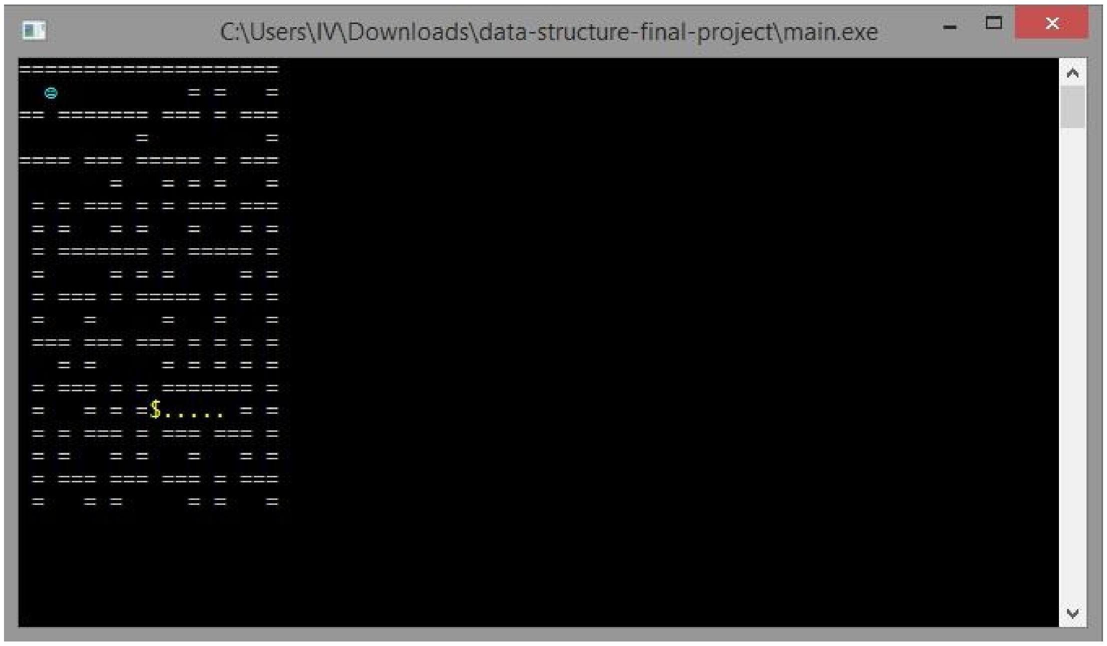
        h) Maze();
        i) Maze(int size);

H. Maze.cpp

1. Implementation of the Maze class from Maze.h
2. void initGrid(): function is to initialize the 2d array of the map
3. void printMaze(): function to print the maze
4. bool inMaze(int x, int y): function to check if the given coordinate (x, y) is in the maze (not out of boundary of the maze)
5. void addWalls(int x, int y): function that is used in the generation of the maze (grid). To add the walls that are available to be traversed for the maze generation
6. void generateMaze(): function to generate the maze of the game (grid) with the Randomized Prim's Algorithm
7. bool isWalkable(int x, int y): function to check if the given coordinate of the maze is walkable or is a passage for the Breadth-First-Search algorithm
8. bool isWalkablePlayer(int x, int y): function to check if the move that the player made is valid
9. int getSize(): function to get the size of the map
10. void setVisited(int x, int y): function to set the given coordinate in the map to the state "visited" for the Breadth-First-Search algorithm.
11. void removeVisited(): function to reset the "visited" state in the map for the Breadth-First-Search algorithm.

VI. Program Manual

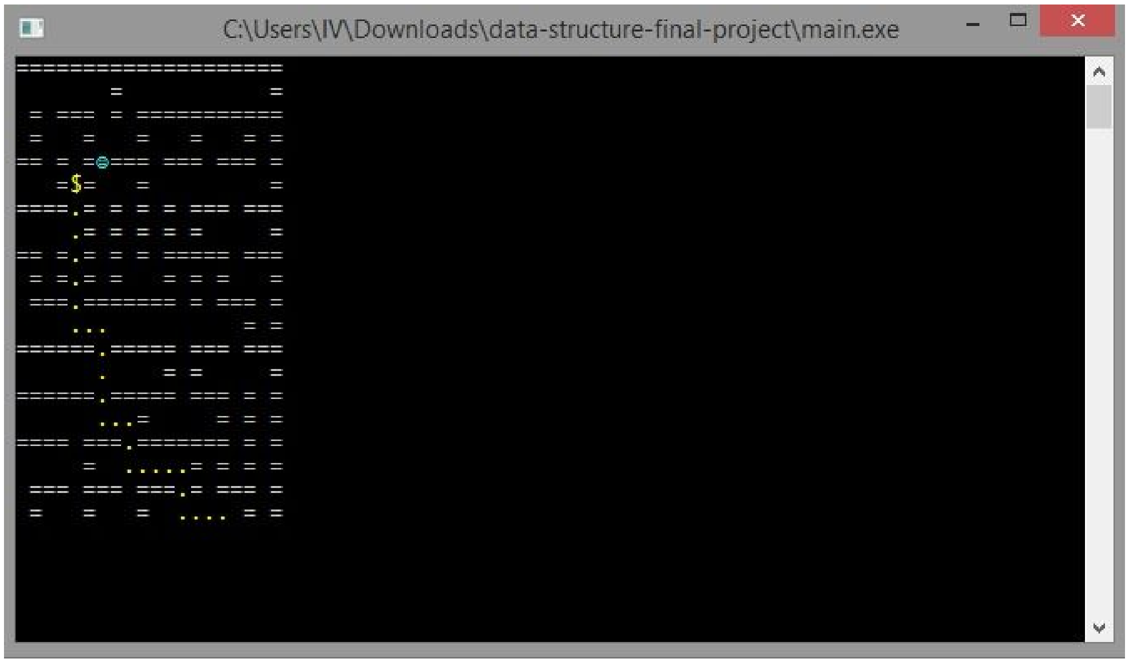
A. First, the player's character is spawned at coordinate (2,1) and the enemy will be spawned at coordinate a random coordinate between (10,15) to (15,10)



C:\Users\IV\Downloads\data-structure-final-project\main.exe

```
====================
  ☺          = =   =
== ======= === = ===
         =         =
==== === ===== = ===
       =   = = =   =
 = = === = = === ===
 = =   = =   =   = =
 = ======= = ===== =
 =     = = =     = =
 = === = ===== = = =
 =   =     =   =   =
 === === === = = = =
   = =     = = = = =
 = === = = ======= =
 =   = = =$..... = =
 = = === = === === =
 = =   = =   =   = =
 = === === === = ===
 =   = =     = =   =
```

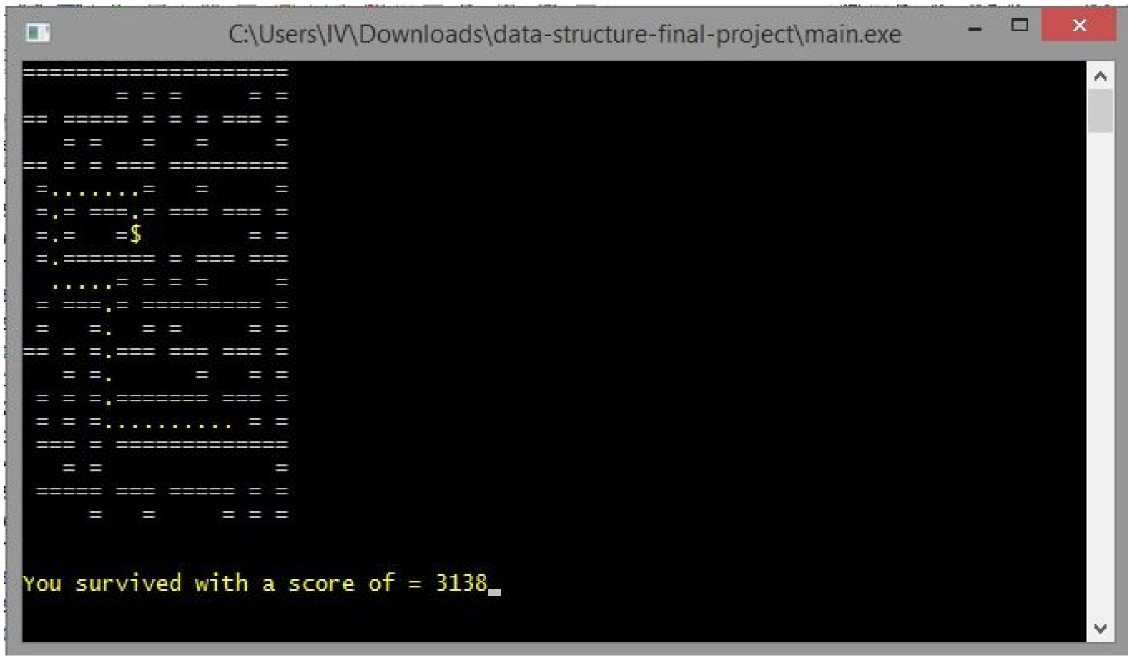
B. Next, the user will then operate the player using the arrow keys up,down,left, and right from the keyboard to escape from the grasp of the enemy.



C:\Users\IV\Downloads\data-structure-final-project\main.exe

```
====================
       =           =
 = === = ===========
 =   =   =   =   = =
== = =☺=== === === =
   =$=   =         =
====.= = = = === ===
    .= = = = =     =
== =.= = = ===== ===
 = =.= =   = = =   =
 ===.======= = === =
    ...          = =
======.===== === ===
      .    = =     =
======.===== === = =
      ...=     = = =
==== ===.======= = =
     =  .....= = = =
 === === ===.= === =
 =   =   =  .... = =
```

C. If the enemy caught the player, the program will terminate and the score will be displayed.



```
====================
       = = =     = =
== ===== = = = === =
   = =   =   =     =
== = = === =========
 =.......=   =     =
 =.= ===.= === === =
 =.=   =$        = =
 =.======= = === ===
  .....= = = =     =
 = ===.= ========= =
 =   =.  = =     = =
== = =.=== === === =
   = =.      =   = =
 = = =.======= === =
 = = =.......... = =
 === = =============
   = =             =
 ===== === ===== = =
     =   =     = = =


You survived with a score of = 3138
```

VII. Git Link

https://github.com/yoksanherlie/data-structure-final-project