בס"ד

**EX2-BOMBERMAN**

Fs891243- Isaac Michaan- יצחק משען

308362425 – Binyamin Goldstein – בנימין גולדשטיין

Description:

The program simulates a computer game called Bomberman, where the robot must leave the factory (to reach the door) without being caught by the guards.

If he is caught then he loses life, and starts again from the beginning of the stage.

The robot can take out a bomb that will be activated after 5 turns. We will see the countdown of the bomb until the explosion. From a programing perspective:

The program is built to support a small interactive calculator, made to allow the user to add & evaluate some mathematical functions at will. Thus - the program is very responsive & its structure support adding new objects according to the user's will.

We have used several classes & files to divide the missions into a logical modules (see 'Design' label below). The program is reading the options info from files, so that it would be easy to replace it by a different layout at will.

Files:

**Robot.h/cpp:**

The part responsible for the robot, check the

1. Location.

2. Move its key

3. Quantity of life.

4. Steps.

5. Placing the bomb.

6. Reset the data in case of disqualification or termination of the robot stage.

7. Calculate points for each step.

**Guard.h/cpp:**

The part responsible for the guards, check out the

1. Their location.

2. Their movement.

3. Check if they do not explode.

**Board.h/cpp:**

The part responsible for the board, check out the

1. Position of the robot.

2. Reboot and position guards.

3. Movement of the robot and guards in turn.

4. Print the board each time.

**Console.cpp:**

The part that is responsible for running the game completely.

From reading to the clipboard, building the pieces to a show until the game restarts.

**Board.txt :**

File stages of the game.

**KeyBoard.h/cpp:**

In charge of the special keys we will not accept the bug I use

**Design:**

We have the game manager (Control) who takes care of all the game arrangements.

There's a robot class that takes care of all the robot's stuff (including the bomb).

There's a ward of guards guarding all the guards' things.

And there's a class of the board that puts it all out on the board.

Main data structures and their roles:

Algorithms that deserve to be highlighted:

Known bugs:

The bomb has a problem after it kills the guard, it sometimes does not change and shows the robot as if it stays in the same place even though it progresses until it takes out a new bomb.

And after he kills the guard, the guard continues to be on the board and move even though he should not be (the problem is the deletion of the guard's vector).

Also, there is a problem (errors) in the KeyBoard file that we did not know how to arrange them (but this file was not implemented by us but we received with the exercise).

Other notes: