ATILIM UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CMPE 113 Computer Programming I
2020 - 2021 Fall
Instructors: Cansu Çiğdem Ekin, Gonca Gökçe Menekşe Dalveren
Assistant: Buğra Yener Şahinoğlu
Homework II
Due Date: 02.05.2021 / 23:59

RULES & REGULATIONS

- You must **submit** your homework **via Moodle before the due date/time**.
- Homework submitted via email will be **ignored**!
- **Late submissions will NOT be graded.**
- **Before submission check that**:
  - Use **CodeBlocks** or **DevC++** as the IDE
  - Name your homework file as in the format **lastname_firstname_hw2.c**.
  - **Do NOT use Turkish characters when you name C source file.**
  - Your homework must be a source file (**a C file** not CPP).
  - **Do NOT upload .exe file. Otherwise you will get ZERO.**
- **Your codes** will be checked by special software (JPLAG & MOSS) for code similarity. If the **code similarity** between any two or more submissions is **higher than %90**, we will also examine and compare these codes by eye. If we are convinced that the similarity between two codes is not merely a coincidence, **all involved HWs will get 0 as the grade**.
- **Therefore**;
  - **Group study is not allowed.** Everyone needs to do his/her homework as an individual.
  - **Do not use Internet resources!** If any other student use the same source then your submissions will be found as similar!
  - **Do not share your solution/ideas with others.**
- Moreover, if we believe that the students cheat on these exams, we will initiate a **disciplinary act**!
- **GRADING POLICY**: If your code can **not run** *at all* or *as expected,* your code will be evaluated over **70 points not 100 points**.

Question

You will write the program of an action-rpg-like game.

In the console, player sees a grid by 20x30 (row x column). In this grid, player’ s character starts in the first cell, and certain amounts of trees and enemies are located in random cells. Each cell are hidden, that means player doesn’t see the enemies or trees. Player only see the cells that are next around the him/her or the cells that has been already passed by and make it visible (fog of war). Player can see the all of the grid with activating the cheat. Pressing **c** activate and deactivate the cheat. If the cheat is active all the cells in the grid will be visible, otherwise all of them will be hidden. Player can move in the grid with using **w-a-s-d** keys. If there is a tree or enemy appear in the path of the player, he/she can’t move to that direction. If the enemy is right next to the player, the player can kill the enemy with pressing **h**. If all the enemies are death or the player presses **q** game ends.

You are going to make this game according to following instructions:

In your program you have two 2-dimensional char arrays one of them called **displayMatrix** other called **placementMatrix**. Both of them should be the same size. Player's position is stored in **a global array** with length 2.
You are going to prepare the displayMatrix with the **prepareDisplayMatrix**() function. It takes the sizes of the matrix and the displayMatrix as parameters. Assign **'P'** to the first cell where player starts and ▓ **(Ascii = -78)** to the others.

Also prepare the placementMatrix with the **preparePlacementMatrix**() function. It takes the sizes of the matrix, the placementMatrix, tree count and enemy count as parameters. Assign each cell a space character **(' ')** first, then select random cells amount of tree count and enemy count parameters and assing **'T'** for trees and **'E'** for enemies. Enemies and trees cannot be generated in the first cell and the cells one next around it. To generate random numbers you can use **rand()** function. **rand()%number** gives you random values between 0 and number. Also you need to add **srand()** function, to generate different sequence of random numbers each time game run, to the main function. To use these functions, you need to add **stdlib.h** and **time.h** libraries to your code.

Call prepareDisplayMatrix() and preparePlacementMatrix() functions inside the **prepareTheGame**() function. It takes size of the matrices, displayMatrix, placementMatrix, tree count and enemy count as parameters.

Write a function called **update**() and call this inside the main function. It takes size of the matrices, displayMatrix, placementMatrix and an integer value to store enemy count as parameters. In this function, you will create a game loop. Inside this loop, get input from the player with using **getch()** function. To use this function, you need to add **conio.h** library to your code. Inside the loop call **fightWithTheEnemy**() function to check if the player fights with the enemy, **updateDisplayMatrix**() function to check if player moves and display the grid with using **display()** function. Inside the loop check if player press **'c'** as an input. If he/she is, activate or deactivate the cheat mode. If cheat mode activated call the **display()** function to show the grid, otherwise call the **displayGodMode**() function to show the grid. Game loop ends when player press **'q'** or all the enemies are killed. Clear the console with using system("cls"); function every time there is a change (or once in every game loop).

fightWithTheEnemy() function takes sizes of the matrix, placementMatrix, Input character and the refence of the integer parameter of the update function as parameters. In this function, first check if the player's Input is **'h'**, if it is then check if there are any enemy in a cells that one next to the player's cell. If there is kill the enemy by changing its value with the space (' ') in the placementMatrix. Player can only one enemy at time.

updateDisplayMatrix() function takes sizes of the matrices, displayMatrix, placementMatrix and the Input character as parameters. In this function, first check the player's movement. If the Input parameters are **w-a-s-d**, player moves according to input value. Player cannot move to a cell if there is an enemy or tree located in it. For each input, player moves only one cell. Then make visible all the cells around the player. If player moves update the global array that stores the player's position, set the character to the 'P' in the cells in that position in the displayMatrix and set the previous cell that player is located to the ▒ **(ascii= -80),** and make all the cells next around the player visible. If there is a space in the cell that found in same index in the placementMatrix, set the cell in the display matrix to ▒ (ascii=-80), otherwise set the cell in the displayMatrix as same as whats in the placementMatrix.

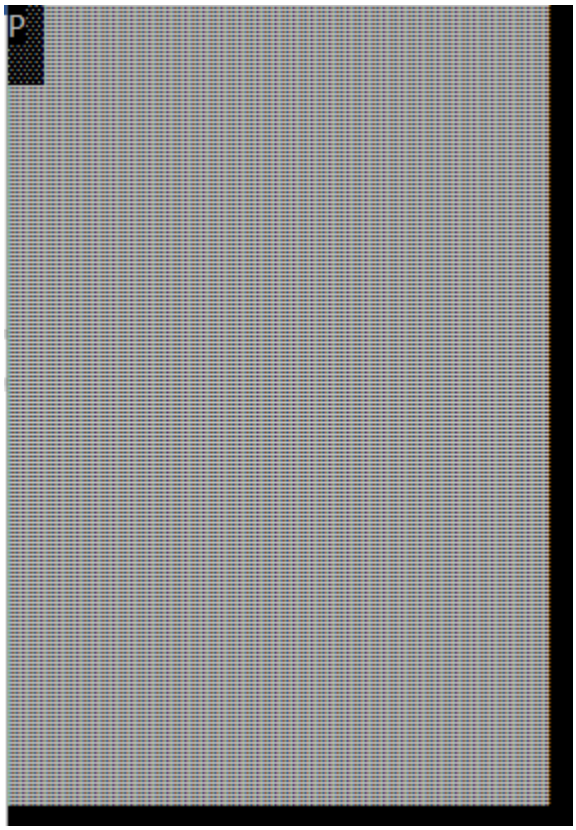
display() function takes sizes of the matrix and displayMatrix as parameters and shows the displayMatrix on the console.

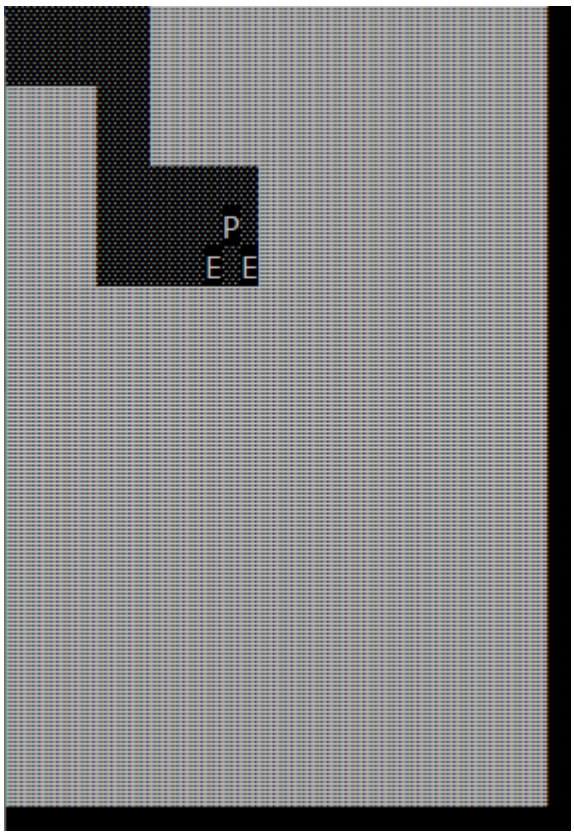
displayGodMode() function takes sizes of the matrices, displayMatrix and placementMatrix as parameters. It shows the display matrix, but if the character in the displayMatrix's cell is ▓ **(Ascii = -78),** it looks to the cell that found in placementMatrix and display it. If the value of the cell in the placementMatrix is space(' '), display ▒ **(ascii= -80).**

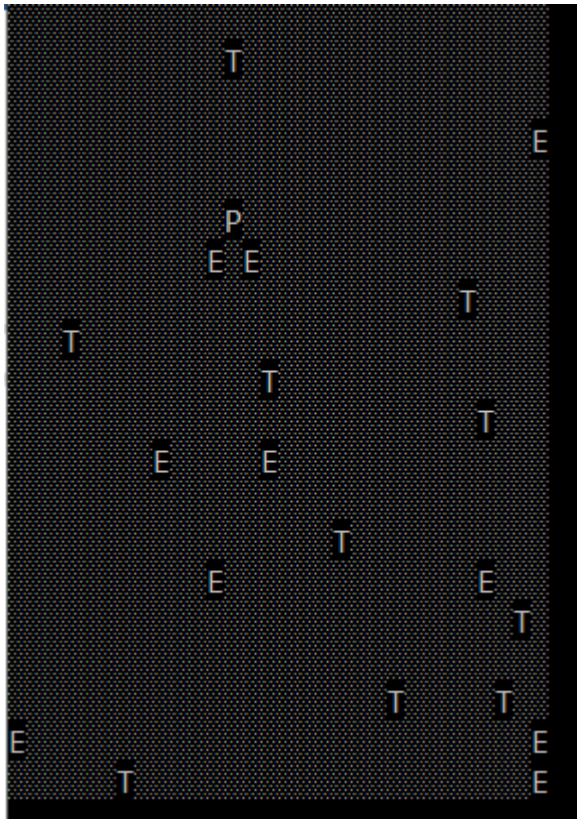
NOTES!:

- You can use scanf() to get an input from the player, but getch() is much more handy, since it does not require pressing the enter key.
- You are free to not clear the console, but if you do that leave some space each time you display the game grid.
- Player can't move outside of the matrix, because that caused the program crashes. You need to check if player try to move outside of the matrix and block the movement. Otherwise it will be accepted as runtime error and your code will be evaluated according to this.
- A video explaining the operation of the game will be added to the Moodle.

Sample Runs:




P
E E


T
E
P
E E
T
T
T
T
E E
T
E E
T
T T
E E
T