MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG140 C Programming Assignment 3

Date handed out: **05 April 2020 Sunday**
Date submission due: **19 April 2020 Sunday 23:55**

Programming Assignment 3: Diamond-Hunter

Purpose:
The main purpose of this programming assignment is to revise the topics that we have covered so far including arrays, pointers, functions, repetitive statements, conditional statements and fundamentals of C programming. In this assignment, you are going to implement a simple one-player game based on the given specifications below. The main goal of this game is to hunt the diamonds that are hidden in a board.

In your implementation you will keep the coordinates of the diamonds inside a two dimensional array. This will help you to practice with arrays, pointers and all of the three control constructs (sequence, selection and repetition). Do not try to compile your entire program in one "big bang". Compile it piece by piece. **Test each function/peice that you have compiled to make sure it works correctly before you add the next function/piece.**

Game Specifications:
Diamond-Hunter is a single player game. There will be one player and once (s)he starts the game (s)he can play it as many times as they like.

The game will start with a UserBoard of 7x7 (7 rows and 7 columns) and a DiamondBoard of 7x7 (7 rows and 7 columns). Only UserBoard will be visible to the user. You will keep DiamondBoard secret (will not be visible to the user). UserBoard will be all initialized with '**?**'. The DiamondBoard will have n diamonds hidden, where n will be entered by the user at the beginning of the game. These n diamonds will be located on the board randomly. Their coordinates will be randomly choosen and set to a diamond '*' on the DimondBoard, while remaining will be set to '**?**'. For example;

Diamond-Hunter
Lets get started!
Enter the number of diamonds to hunt: 6

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	?	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?
7	?	?	?	?	?	?	?

DiamondBoard

	1	2	3	4	5	6	7
1	?	?	*	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	*	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	?	?	?
7	?	?	?	?	*	?	?

Once the game is started as shown in the above example, then the user can play on this UserBoard at most 10 rounds. If the user decides to play the game again, then the program will show the user the highest points accumlated so far and user can play again. Hence, the player can play as many games as they like but one game can consists of at most 10 rounds.

The game will be played as follows. The user will try to find as many diamonds as possible. In order to find a diamond, the user will specify the guessed location in the UserBoard by the following format: (row, column). For example, the program will ask:

```
Round 1:
Enter the coordinates of the diamonds: (3,2)
You got 100 points!
Your total points is 100!
```

Program will check if the corresponing row and column of DiamondBoard (where in the specific example, DiamondBoard[3][2]) contains a diamond or not. If it does contain a diamond (where in the specific example, it does), then program will change that element of UserBoard (where in the specific example, UserBoard[3][2]) to '*' and user will gain 100 points. Please note that the board coordinates start from 1 but in actual coding, you need to start ofcourse from 0.

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?
7	?	?	?	?	?	?	?

DiamondBoard

	1	2	3	4	5	6	7
1	?	?	*	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	*	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	?	?	?
7	?	?	?	?	*	?	?

Otherwise, for example;

```
Round2:
Enter the coordinates of the diamonds: (5,5)
You did not get any points!
Your total points is 100!
```

Program will check if the corresponing row and column of DiamondBoard (where in the specific example, DiamondBoard[5][5]) contains a diamond or not. If it does not contain a diamond (where in the specific example, it does not), then program will count the number of diamonds around the 8 neighbours of that index in the DiamondBoard (where in the specific example, 8 neighbors of [5][5] are: [5][4], [4][4], [4][5], [4][6], [5][6], [6][6], [6][5], [6][4]) and change that element of UserBoard (where in the specific example, UserBoard[5][5]) to a count of diamonds, and user will get 0 points. By this way, user can obtain hint about how many diamonds are there around 8 neighbours of that cell!

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	2	?	?
6	?	?	?	?	?	?	?
7	?	?	?	?	?	?	?

DiamondBoard

	1	2	3	4	5	6	7
1	?	?	*	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	*	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	?	?	?
7	?	?	?	?	*	?	?

This will continue at most 10 rounds . If user can not find all diamonds within these constraints, (s)he will lose the game. Otherwise, whenever user finds all diamonds, (s)he will win and program will stop asking for more selection. Then, the program will show user his/her total points and highest points and will ask them if they want to play again. They can play as many times as they like.

Programming Requirements:

In order to implement this game you will need to write at least the following functions, but if you need more functions you can add them.

Function	**Explanation**
`main`	This needs to create the Userboard and DiamondBoard, gets the value of n (number of diamonds) from the user, maintain the current total points and the highest point of the user, calls the function `initializeBoards` to initialise boards, calls the function `getGuessedCoordinates` that takes the coordinates from the user, calls the function `checkDiamonds` to verify that user guess is a diamond or not, and uses the function `provideHints` when the user guess is not a diamond, and maintains the points accordingly and finally, asks the user if they want to play the game again. This function also uses the `display_board` function to show the status of the UserBoard to the user after each play. See the sample run!
`initializeBoards`	This function will get the Userboard populate it with '**?**' and randomly populates Diamondboard with n '*****' and remaining with '**?**'. In order to randomly generate coordinates of '*****' it will use the `randomDiamond` function twice, one for the row and one for the column . **Input**: an array of 7x7, an array of 7x7 and n. **Output**: No output
`randomDiamond`	This function will randomly generate a random number that will represent a diamond's coordinate. **Input**: none **Output**: one integer number, which can represent a row or column

	of a diamond.
`getGuessedCoordinates`	This function will ask user to enter the coordinates in the format of "(row column)". **Input**: row and column variables (row, column). **Output**: integer that shows if the operation is successfully completed.
`checkDiamonds`	This function will take the coordinates of the user guess and then will return if it is a diamond or not. **Input**: row, column and Diamondboard **Output**: true if it is a diamond and false if it is not diamond.
`provideHints`	This function will count the number of diamonds at the eight neighbours of the given coordinate using the DiamondBoard. Then, will change the '**?**' in the UserBoard at the given coordinate with the count of diamonds. **Input:** row, column, UserBoard and Diamondboard **Output:** successful or not
`displayUserBoard`	This function will clear screen and display the current status of the UserBoard with the current positions of the player.

Assume DiamondBoard is initialised as follows;

DiamondBoard

	1	2	3	4	5	6	7
1	?	?	*	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	*	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	?	?	?
7	?	?	?	?	*	?	?

Then Sample Run could be as follows:

```
*Diamond-Hunter*
Lets get started!
Enter the number of diamonds to hunt: 6
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	?	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?
7	?	?	?	?	?	?	?

```
Round 1:
Enter the coordinates of the diamonds: (7,4)
You got 0 points!
Your total points is 0!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	?	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?
7	?	?	?	2	?	?	?

```
Round 2:
Enter the coordinates of the diamonds: (7,3)
You got 0 points!
Your total points is 0!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	?	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?
7	?	?	1	2	?	?	?

```
Round 3:
Enter the coordinates of the diamonds: (3,2)
You got 100 points!
Your total points is 100!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	?	?	?	?
7	?	?	1	2	?	?	?

```
Round 4:
Enter the coordinates of the diamonds: (6,4)
You got 100 points!
Your total points is 200!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	*	?	?	?
7	?	?	1	2	?	?	?

```
Round 5:
Enter the coordinates of the diamonds: (6,5)
You got 0 points!
Your total points is 200!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	*	2	?	?
7	?	?	1	2	?	?	?

```
Round 6:
Enter the coordinates of the diamonds: (7,5)
You got 100 points!
Your total points is 300!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	?
5	?	?	?	?	?	?	?
6	?	?	?	*	2	?	?
7	?	?	1	2	*	?	?

```
Round 7:
Enter the coordinates of the diamonds: (4,7)
You got 100 points!
Your total points is 400!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	2	?	?
7	?	?	1	2	*	?	?

```
Round 8:
Enter the coordinates of the diamonds: (1,3)
You got 100 points!
Your total points is 500!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	*	?	?	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	2	?	?
7	?	?	1	2	*	?	?

```
Round 9:
Enter the coordinates of the diamonds: (1,5)
You got 0 points!
Your total points is 500!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	*	?	0	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	?	?
4	?	?	?	?	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	2	?	?
7	?	?	1	2	*	?	?

```
Round 10:
Enter the coordinates of the diamonds: (3,6)
You got 0 points!
Your total points is 500!
```

UserBoard

	1	2	3	4	5	6	7
1	?	?	*	?	0	?	?
2	?	?	?	?	?	?	?
3	?	*	?	?	?	1	?
4	?	?	?	?	?	?	*
5	?	?	?	?	?	?	?
6	?	?	?	*	2	?	?
7	?	?	1	2	*	?	?

```
You run out of rounds! Game over!
Would you like to play again (Y/N)?
```

Grading Policy:
Your program will be graded as follows:

Grading Point	Mark (100)
`Main`	20
`initializeBoards`	10
`randomDiamond`	10
`getGuessedCoordinates`	10
`checkDiamonds`	15
`provideHints`	15
`displayUserBoard`	10
Code quality (Appropriate comments, variable names, formulation of selection statements and loops, reusability, extensibility etc.)	10

Please make sure that you follow the restrictions for the assignment as follows.

- **You are not allowed to use global variables**.
- Strictly obey the input output format. Do not print extra things.
- You are not allowed to use `goto` statement.
- Name your source file "StudentID.c"
- Upload only source file. Do not compress it (zip, rar, …)