Coding Assignment 10

Create a #define called MAPSIZE. Set to 10.

Create global variable `MapFilename`. Use `char` array and set them to nulls by using {}. Make it at least 20.

Create a structure called `PlayerInfo` that has a `char` array called `PlayerName` of size 20 and two `int` members called `CurrentRow` and `CurrentCol` to hold the current position of the player.

Function `MoveNorth` should have a return value of `int` and two parameters - the `Map` array and a pointer to `Player`.

```
{
        If moving the Player's current position north results in a spot on the map that is part of the path (Map element is set to
        X, S or E), then decrement the Player's row and return true to show that a move was made; otherwise, print message
        "North is the wrong move" and return false to show that a move was not made.
}
```

Function `MoveSouth` should have a return value of `int` and two parameters - the `Map` array and a pointer to `Player`.

```
{
        If moving the Player's current position south results in a spot on the map that is part of the path (Map element is set to
        X, S or E), then increment the Player's row and return true to show that a move was made; otherwise, print message
        "South is the wrong move" and return false to show that a move was not made.
}
```

Function `MoveEast` should have a return value of `int` and two parameters - the `Map` array and a pointer to `Player`.

```
{
        If moving the Player's current position east results in a spot on the map that is part of the path (Map element is set to
        X, S or E), then increment the Player's colum and return true to show that a move was made; otherwise, print message
        "East is the wrong move" and return false to show that a move was not made.
}
```

Function `MoveWest` should have a return value of `int` and two parameters - the `Map` array and a pointer to `Player`.

```
{
        If moving the Player's current position west results in a spot on the map that is part of the path (Map element is set to
        X, S or E), then decrement the Player's column and return true to show that a move was made; otherwise, print
        message "West is the wrong move" and return false to show that a move was not made.
}
```

Function `get_command_line_params` should have a return value of `void` and parameters of `argc` and `argv`

```
{
        Create a for loop with argc to go through each value of argv
                Look for an argv of MAP=
                        When found, take the string after the = as the value to store in MapFilename.
        After looking at all of the strings in argv, if you do not have a value for MapFilename, then print the message
                "MAP= must be given on the command line"
        and exit.
}
```

```
main()

{

    Declare one variable of type FILE * called TreasureMap

    Create a char array called MapList of size 500

    Create a 2D char array called Map.  The array should be MAPSIZE for both dimensions (it is square).

    Create a 2D char array called PlayerPath.  The array should be MAPSIZE for both dimensions (it is square).

    Create a char variable named PlayerMove of size 1.  This will hold the direction move entered by the player.

    Create int variables i, j and k.  These will be used later in for loops.

    Create int variable named MakeMove.  This will hold the return value of the direction functions.

    Declare a variable called Player of type struct PlayerInfo.

    Declare a pointer to Player called PlayerPtr and initialize it to the address of Player.

    Call function get_command_line_params

    Pass MapFilename to fopen() with a mode of r+ and store the return value in TreasureMap.

    Check if TreasureMap is NULL.  If it is, call perror() with "TreasureMap did not open" and exit.

    Clear the screen

    Prompt user for name "Enter Player name". scanf() with %s to put value in Player.PlayerName.  Call
    getchar() to get rid of <ENTER>

    Set Player.CurrentRow to 0 and Player.CurrentCol to 0

    Use fgets() to read 500 characters from TreasureMap into MapList

    /* Initialize the map to the values found in TreasureMap and initialize PlayPath to dashes */

    Set i = 0

    Create a for loop from j = 0 to j < MAPSIZE
    {
        Create a for loop from k = 0 to k < MAPSIZE (this is nested inside the j for loop)
        {
            Set Map[j][k] to MapList[i]
            Print Map[j][k] to the screen
            Increment i
            Set PlayerPath[j][k] to a dash
        }
        Print a blank line
    }

    Print "Press <ENTER> when you are ready to start "
    Use getchar() to pause and capture <ENTER> when pressed.
    Clear the screen


    Set PlayerPath[0][0] to Map[0][0]
    do
    {
        Print "Enter a direction (NSEW) or 'Q' to quit "
        scanf() entered direction into PlayerMove
        Uppercase PlayerMove
        Clear the screen
        if PlayerMove is 'N', then call MoveNorth() and pass Map and PlayerPtr
            MakeMove will receive the return value of MoveNorth().
        else if PlayerMove is 'S', then call MoveSouth()and pass Map and PlayerPtr
            MakeMove will receive the return value of MoveSouth().
        else if PlayerMove is 'W', then call MoveWest()and pass Map and PlayerPtr
            MakeMove will receive the return value of MoveWest().
        else if PlayerMove is 'E', then call MoveEast()and pass Map and PlayerPtr
            MakeMove will receive the return value of MoveEast().
        else
            Print "Invalid move - must be NSEW" and set MakeMove to 0

        Set Map[Position.row][Position.col] to 'X' to signal a move to this position

        if MakeMove is TRUE
            Don't overwrite 'S' or 'E'.
            Find the Player's current position in Map.
            If that location in Map is 'S', then don't update PlayerPath and print "Player x is back at start".
            If that location in Map is 'X', then update the same location in PlayerPath with 'X'.
            If that location in Map is 'E', then print "Player x has made it to the end - WINNER!!".  Update the current position in PlayerPath to 'E' and set PlayerMove to 'Q'.

        Print PlayPath to the screen to show the Player their moves.

    }
    while PlayerMove is not 'Q'

    Call fclose() with TreasureMap to close it

}
```

How to Run

1. Run your Code9.c with a directions file to create a map file

```
a.out DIRECTIONS=direction2 MAP=map2
```

In this example, `direction2` contains

```
[frenchdm@omega ~]$ more direction2
ESESESESESES
```

Running the program results in

```
S   X   -   -   -   -   -   -   -   -

-   X   X   -   -   -   -   -   -   -

-   -   X   X   -   -   -   -   -   -

-   -   -   X   X   -   -   -   -   -

-   -   -   -   X   X   -   -   -   -

-   -   -   -   -   X   X   -   -   -

-   -   -   -   -   -   E   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -
```

and this MAP file

```
[frenchdm@omega ~]$ more map2

SX---------XX---------XX---------XX---------XX---------XX---------E----------
----------------------
```

Now run `Code10.c` using the map output from `Code9.c` as input to `Code10.c`

```
     a.out MAP=map2
```

The screen will clear and then the map will print. Player is given as much time as they want to study the map. When they press <ENTER>, the game begins.

```
Enter Player Name : Fred
S   X   -   -   -   -   -   -   -   -

-   X   X   -   -   -   -   -   -   -

-   -   X   X   -   -   -   -   -   -

-   -   -   X   X   -   -   -   -   -

-   -   -   -   X   X   -   -   -   -

-   -   -   -   -   X   X   -   -   -

-   -   -   -   -   -   E   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -
```

```
Hi Fred!  Press <ENTER> when you are ready to start
```

Pressing <ENTER> clears the screen and the Player is prompted for a direction

```
Enter a direction (NSEW) or 'Q' to quit
```

Entered direction is tested against MAP to see if the entered move is valid. If it is not, then a message prints, the player's progress prints and the player is prompted for a new direction.

```
Wrong direction - try again
S   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

Enter a direction (NSEW) or 'Q' to quit
```

If the entered move is valid, the player is told that it is correct and the player's progress prints again

```
East is the correct move
S   X   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -
```

```
Enter a direction (NSEW) or 'Q' to quit
```

This process continues until the player finds the end.

```
South is the correct move
Player Fred has made it to the end - WINNER!!

S   X   -   -   -   -   -   -   -   -

-   X   X   -   -   -   -   -   -   -

-   -   X   X   -   -   -   -   -   -

-   -   -   X   X   -   -   -   -   -

-   -   -   -   X   X   -   -   -   -

-   -   -   -   -   X   X   -   -   -

-   -   -   -   -   -   E   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -

-   -   -   -   -   -   -   -   -   -
```