Ahmad Mohammad

c1001633 / asm6t

proj4

		deduction
TURNIN TIME	turned in on time.	0 %
SOURCE CODE SEARCH RESULTS	MISSING: None. FOUND: TREASUREMAP_H, treasuremap.h, class TreasureMap, TreasureMap::TreasureMap, TreasureMap::FindTreasure, TreasureMap::PrintMap, TreasureMap::getMove, make_pair	0

DEDUCTIONS	-0
FINAL GRADE	100

COMPILATION LOG

```
c1001633 MAKE PASS

MAKELOG OF STUDENT PROVIDED MAKEFILE:
g++ -c -g -O0 -std=c++11 -Wall treasuremap.cpp -o treasuremap.o -MMD -MF treasuremap.d
g++ -c -g -O0 -std=c++11 -Wall proj4.cpp -o proj4.o -MMD -MF proj4.d
g++  -o proj4 treasuremap.o proj4.o
treasuremap.cpp: In member function ‘std::pair<int, int> TreasureMap::getMove(char)’:
treasuremap.cpp:97:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
```

```
c1001633 runlog begin:
Test 1:
Inputs:
NESW
4 5
LLL~~
~L~~L
~LLLL
~~LL~
```

✓

```
Output:
```

```
Current map:
L L L ~ ~
~ L ~ ~ L
~ L L L L
~ ~ L L ~
Start Position: 1,1
Searching 1,1
Current map:
* L L ~ ~
~ L ~ ~ L
~ L L L L
~ ~ L L ~
Searching 1,2
Current map:
* * L ~ ~
~ L ~ ~ L
~ L L L L
~ ~ L L ~
Searching 1,3
Current map:
* * * ~ ~
~ L ~ ~ L
~ L L L L
~ ~ L L ~
Searching 2,2
Current map:
* * * ~ ~
~ * ~ ~ L
~ L L L L
~ ~ L L ~
Searching 3,2
Current map:
* * * ~ ~
~ * ~ ~ L
~ * L L L
~ ~ L L ~
Searching 3,3
Current map:
* * * ~ ~
~ * ~ ~ L
~ * * L L
~ ~ L L ~
Searching 3,4
Current map:
* * * ~ ~
~ * ~ ~ L
~ * * * L
~ ~ L L ~
Searching 3,5
Current map:
* * * ~ ~
~ * ~ ~ L
~ * * * *
~ ~ L L ~
Searching 2,5
```

```
Current map:
* * * ~ ~
~ * ~ ~ *
~ * * * *
~ ~ L L ~
Searching 4,4
Current map:
* * * ~ ~
~ * ~ ~ *
~ * * * *
~ ~ L * ~
Searching 4,3
Current map:
* * * ~ ~
~ * ~ ~ *
~ * * * *
~ ~ * * ~
No Treasure!

Test 2:
Inputs:
SNEW
5 4
L~~~
LLL~
~LL~        ✓
T~L~
LLLL

Output:
Current map:
L ~ ~ ~
L L L ~
~ L L ~
T ~ L ~
L L L L
Start Position: 1,1
Searching 1,1
Current map:
* ~ ~ ~
L L L ~
~ L L ~
T ~ L ~
L L L L
Searching 2,1
Current map:
* ~ ~ ~
* L L ~
~ L L ~
T ~ L ~
L L L L
Searching 2,2
Current map:
* ~ ~ ~
* * L ~
```

```
~ L L ~
T ~ L ~
L L L L
Searching 3,2
Current map:
* ~ ~ ~
* * L ~
~ * L ~
T ~ L ~
L L L L
Searching 3,3
Current map:
* ~ ~ ~
* * L ~
~ * * ~
T ~ L ~
L L L L
Searching 4,3
Current map:
* ~ ~ ~
* * L ~
~ * * ~
T ~ * ~
L L L L
Searching 5,3
Current map:
* ~ ~ ~
* * L ~
~ * * ~
T ~ * ~
L L * L
Searching 5,4
Current map:
* ~ ~ ~
* * L ~
~ * * ~
T ~ * ~
L L * *
Searching 5,2
Current map:
* ~ ~ ~
* * L ~
~ * * ~
T ~ * ~
L * * *
Searching 5,1
Current map:
* ~ ~ ~
* * L ~
~ * * ~
T ~ * ~
* * * *
Searching 4,1
Treasure Found!
```

```
Manifest: proj4.cpp.lst
-----------------------
-----------------------
 Mar 17 18:15 proj4.cpp
// Ahmad Mohammad
// CSCI 3110-001
// Proj 4 Implementation file for TreasureMap class/header
// Due: 03/17/22
// DESCRIPTION: This program has the capabiltiy to find its way to a "treasure" (T) from
// a starting point on an island (array) using recursion.

#include "treasuremap.h"
#include <iostream>


using namespace std;

int main()
{
        // var declaration and print initial map
        TreasureMap t1;
        t1.PrintMap();
        bool x = false;
        int i = 1;
        int y = 1;
        cout << "Start Position: " << i << "," << y << endl;
        // start recursion with starting on 1,1
        t1.FindTreasure(i,y,x);

        if (x == true)
        {
                cout << "Treasure Found!" << endl;
        }
        else
                cout << "No Treasure!" << endl;

        // end of prog
        return 0;
}
```

✓

```
---treasuremap.h: ----------------
#ifndef TREASUREMAP_H
#define TREASUREMAP_H
#include <iostream>
#include <string>


class TreasureMap                                               // Represents Treasure Map class’ data and function members
{
public:
        TreasureMap();                                          // Constructor: reads and builds treasure map from file
        void PrintMap();                                        // Displays the treasure map and its state
        void FindTreasure(int, int, bool&);     // Recursive function that finds the treasure (ints: row, then column)
private:
        char tmap[10][10];                                      // 2D array of treasure map - outer columns and rows not traversable
        int maxRows;                                            // Maximum number of rows in play area - excludes boundaries
        int maxCols;                                            // Maximum number of columns in play area - excludes boundaries
        std::string xplor;                                      // 4-element string dictating order of map exploration (see specifications)
        std::pair<int,int> getMove(char);       // returns a pair (row, then column offset from current cell)
};

#endif

---treasuremap.cpp: ----------------
// Ahmad Mohammad
// CSCI 3110-001
// Proj 4 Implementation file for TreasureMap class/header
// Due: 03/17/2

#include "treasuremap.h"
#include<iostream>
#include<fstream>

using namespace std;

// default consturctor of TreasureMap object
TreasureMap::TreasureMap()
{
        // declaration of input file
        ifstream infile;
        infile.open("treasuremap.txt");

        // assigns first like of infile to ’xplor’ string (NESW)
        getline(infile, xplor);

        // assigns next two data in the file to row and col variables
        infile >> maxRows;
        infile >> maxCols;

        // declaration of holder character variable
        char space;

```

```
    // for loop to traverse and add X to every index of arr
    for(int i = 0; i <= maxRows+1; i++)
    {
      for(int k = 0; k <= maxCols+1; k++)
      {
          tmap[i][k] = 'X';

      }
    }

    // for loop to replace X with char from infile
        for(int i = 1; i <= maxRows; i++)
        {
            for(int k = 1; k <= maxCols; k++)
            {   infile >> space;
                    tmap[i][k] = space;

            }
        }
}

// function that prints array out with out X borders
void TreasureMap::PrintMap()
{
    cout << "Current map:"<<endl;
  for(int i = 1; i <= maxRows; i++)
  {   for(int k = 1; k <= maxCols; k++)
  {       cout << tmap[i][k]<< " "; }
            cout << endl;
    }

}


pair<int,int> TreasureMap::getMove(char d)
{
    //declaring pair of ints to 'move'
    pair<int, int>  move;


    //if statwments that decide which operation to perform


    if( d == 'N' )
    {
        move = make_pair(-1 , 0);
        return move;
    }
    if( d ==  'E' )
    {
        move = make_pair(0 , 1);
    return move;
  }
    if( d ==  'S' )
  {
    move = make_pair(1 , 0);
    return move;
  }
    if( d ==  'W' )
  {
    move = make_pair(0 , -1);
```

```
        return move;
    }





}

//void recursive function that explores the map.
void TreasureMap::FindTreasure(int row, int col, bool &yesno)
{
        // declaration of row/col to be sent in recursion func
        int newrow, newcol;
        // checks if index is currently not on treasure
        if (tmap[row][col] != 'T' && yesno == false)
        {
                // makes sure index os on Land square
                if (tmap[row][col] == 'L')
        {
                        // assign checked 'L' square to '*' and print new map
                        cout << "Searching "<<row<<","<<col<<endl;
            tmap[row][col] = '*';
                        PrintMap();

                        // loop through each direction in xplor var order
                        for(int i = 0;  i < 4; i++)
                        {
                                        // assign newrow/newcol to row/col before we make index change
                    newrow = row;
                    newcol = col;
                                        // index change
                    newrow += getMove(xplor[i]).first;
                            newcol += getMove(xplor[i]).second;
                                        // recurse until invalid state then default backtrack+continue
                        FindTreasure(newrow,newcol,yesno);

                        }
                }
        }
        // Makes sure we on treasure then end  of func
        if (tmap[row][col] == 'T')
    {
        cout << "Searching "<<row<<","<<col<<endl;
        yesno = true;
    }

}
```