Assignment #8

Introduction to C Programming – COP 3223

Objectives

1. To learn how to design and implement functions for program development
2. To reinforce how to use pass by value and pass by reference variables
3. To learn how to use structures to store information and solve problems

Introduction: Pirate Time

Your friend has not stopped talking about how cool pirates and how awesome it would be if they could be a real-life pirate captain! To amuse your friend, you have decided to create a series of programs about pirates.

Problem: The Hunt is On (treasurehunt.c)

Your friend has arrived with their crew at the Caribbean island on the old pirate's map. It's time at last to dig up some treasure! As captain, your friend will need to direct their crew in where to dig. There's an important catch about the island: it's only accessible for a few hours a day and most of the treasure is buried under the sand. In order to maximize their treasure haul, your friend and their crew will need to carefully manage their time and energy.

Program Details

Your friend has a crew of four pirates. Each pirate will only be able to dig a certain amount each hour before they need to rest. To get to the treasure, a pirate will need to clear all of the sand above it. Then, each pirate is only able to carry a certain amount of treasure, but the treasure will need to be taken back to the ship.

Your program will simulate the few hours available for accessing the island. The island has been split into a 3x3 grid of digging spaces. The captain will track how much sand is covering the treasure in each space on their map. For each hour, the captain will need to choose which section of the area to send each of the pirate crew members. As each pirate is deployed, your program should update the map for the captain. Once all the sand is removed from a section, you can claim the treasure!

Consider the following example:

4 Sand **2 Treasure**	1 Sand 1 Treasure	3 Sand 1 Treasure
1 Sand 1 Treasure	2 Sand 1 Treasure	3 Sand 1 Treasure
1 Sand 1 Treasure	5 Sand 3 Treasure	1 Sand 1 Treasure

There are 4 cubic feet of sand and 2 treasures in the upper left corner.

Your friends crew might look like this:

0	1	2	3
Dig: 3 Carry: 1	Dig:2 Carry: 5	Dig: 4 Carry: 1	Dig: 1 Carry: 3

Crew member 2 will be able to completely clear the sand in one hour. In the next hour, however, the captain should send a different pirate to that square because Crew member 2 cannot carry both of the treasures in that area back to the ship.

Your program should prompt your friend for instructions for each crew member. Ask which section they are being sent to. Let them know if there is still sand in that section. If there is sand, the crew member must dig. Once all the sand is gone, the treasure can be picked up. A crew member can only transport a number of treasures equal to his or her carry value.

Your program should run until either all the sections are empty of treasure or until 8 hours has elapsed. At that point in time, the crew must evacuate the island or be lost at sea! Before closing the program, tell the user how much treasure they were able to obtain.

Implementation Restrictions

You must use the following structures to store information:

```
struct pirate {
      int dig;
      int carry;
};

struct map {
      int sand;
      int treasure;
};
```

It is not sufficient to simply have the structures in your program, you must use them store information and process operations for the program. You may use type definition if you prefer.

Though function prototypes will not be provided, it is expected that you follow good programming design and create several functions with well-specified tasks related to the solution of this problem. Make sure to pay very careful attention to parameter passing.

Hints

Create a 3x3 array for the map. The value sand indicates how much sand is currently covering the treasure. When this value has been reduced to zero, the treasure can be picked up. The value treasure indicates how many treasures are in that location. Both values will be provided by an input file. See below for specifications.

Create a 1x4 array for your pirates. These values will also be provided by an input file. Consider making an initialization function to process this file and initialize your data structures for the program.

Input Specification

The input file will contain 9 pairs of integers representing the sand and treasure values (in that order) for each square. This will be followed by 4 pairs of integers representing the dig and carry values (in that order) for the pirates. Here is the sample file that matches the above example:

```
4 2     1 1     3 1
1 1     2 1     3 1
1 1     5 3     1 1
3 1
2 5
4 1
1 3
```

Remember to prompt the user for the name of the input file and open the file that they specify.

The user will input directions in a 1-based, left to right fashion:

```
1 1     1 2     1 3
2 1     2 2     2 3
3 1     3 2     3 3
```

Output Specification

At the start of each hour, let the captain know how many hours are remaining - "You have X hours left to dig up the treasure." – and show them the status of their pirates:

```
Crew    Dig     Carry
1       X1      Y1
2       X2      Y2
3       X3      Y3
4       X4      Y4
```

Then, for each pirate, show the captain the current state of the map:

```
4s      1s      3s
1s      2s      3s
1s      5s      1s
```

(example initial map; use s to indicate sand is still present)

```
2T      1s      3s
1s      1T      3s
1s      5s      1s
```

(player has removed sand from sections 1 1 and 2 2; use T to indicate treasure is available)

```
-       1s      3s
1s      1T      3s
1s      5s      1s
```

(player has removed both sand and treasure from section 1 1; use - to indicate nothing is left)

Then prompt the user with their crew member number for where to send them:

> Where would you like to send crew member X?

Immediately resolve this by sending the pirate to the corresponding section on the map and update either the sand or treasure value accordingly:

> You have removed some of the sand from this section.
>
> You have removed all the sand from this section!
>
> You take all of the treasure back to the ship!
>
> You take some of the treasure back to the ship.
>
> This section has already been cleared.

If the player claims all of the treasure in all of the sections, print:

> All of the pirate's treasure belongs to you now!

If all of the player must retreat due to time, print:

> You are forced to evacuate the island. You have claimed X pieces of the pirate's treasure!

Sample Runs
Files for sample runs will be available on the webcourse.

Deliverables
One source file: *treasurehunt.c* for your solution to the given problem submitted over WebCourses.

Restrictions
Although you may use other compilers, your program must compile and run using Code::Blocks. Your program should include a header comment with the following information: your name, course number, section number, and assignment title. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details
Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.