Lancaster University

SCC.110 Software Development

Problem Set 9: Assessed Exercise

Introduction

> *Twas the night before Christmas, when all through the house*
> *Not a creature was stirring, not even a mouse.*
> *The stockings were hung by the chimney with care,*
> *In hopes that St Nicholas soon would be there.* (Clement Clarke Moore)

Christmas is coming and this year Santa is experimenting with the use of drones to cope with the increased number of parcels to deliver. The elves have constructed a prototype: “Santa’s Latest Electronic Delivery system v1” or SLED1 for short. However, while elves are very good at making toys their software skills are limited - and this is where you come in to help save Christmas.

In this assessed exercise we would like you to write the control software for SLED1. The software needs to read in two files, a mapfile called ‘`map.txt`’ and a command file called ‘`sledcommands.txt`’ (make sure you save both files into the same directory as your program) and use the instructions in the command file to navigate SLED1 around the map.

To read the files you might want to use a new function, `fscanf()`. This is similar to `scanf` expect that it takes as its first parameter *a pointer to a file* to read from (rather than just reading from the keyboard). `fscanf()` returns the number of matching elements that it read. For example, the following code fragment reads in a line from a file pointed to by fp consisting of two integers separated by a ‘`:`’ character. The integers are stored in `i1` and `i2` and result contains the number of items read in.

```
result = fscanf(fp, "%d:%d", &i1, &i2);
```

The Map File

To help SLED1 navigate it stores a map of the local area in a two-dimensional array declared as `char map[30][30]` and initialised so there is the character '`.`' in each location. The coordinates 0,0 are *at the bottom left* of the map and the coordinates 29,29 are *at the top right*. The x-axis runs horizontally and the y-axis vertically.

The map file file consists of 0 or more lines, each of which has the following format.

```
x-cordinate,y-cordinate,symbol
```

x-cordinate, y-cordinate are represented as integers symbol is a character that can be one of:

```
'.' (empty),
'P' (parkland),
'S' (street),
'H' (house),
'T' (tower),
'F' (factory),
'U' (university).
```

An example line might be as follows:

```
9,9,H
```

This indicates that at the coordinate 9,9 there is a house.

The command file consists of 0 or more lines, each of which has the following format.

```
command command_value
```

`command` is a string that can initially be either `FORWARD` or `TURN`
`command_value` is an integer

An example line might be as follows:

```
FORWARD 9
TURN 90
FORWARD 9
```

This set of commands tells SLED1 to move 9 spaces forward on the map, turn 90 degrees clockwise and then move a further 9 spaces. So, for example if SLED1 starts at the coordinates 0,0 then at the end of this command sequence it will be at location 9,9 at which there is a house. SLED1 can only make turns of 0, 90, 180 or 270 degrees.

The Program

Your task is to write a program that can navigate SLED1 around the map specified in the map file according to the instructions supplied in the command file. You should store all the information about SLED1 in a structure of the following type:

```
struct sled {
    int  x,
         y,
         orientation,
         totalDistance;
};
```

This structure needs to be declared *exactly* as above and should appear in your code immediately after any `#includes` or `#defines` but *before* the start of any functions.

Important points:

1. The program should be **your own work**—this is *not a group exercise!* - do not share all or part of your solution with anyone, and do not post it online (*not even to github!*). You *are required* to submit your code for plagiarism checking.
2. If you need help with your programming please seek help from the tutors or the lab demonstrators - *not from colleagues or the Internet*.
3. You *must attend in-person* **your specified** practical class in week 10. You will *only be able to access the questions during your practical class* in week 10.
4. You will need a working program **by the start of week 10** *so please be sure to join the practical class with a working program*. You will be required to modify your program in the lab in week 10.
5. Ensure you read the specification very carefully and that your program *produces exactly the output specified*.

Part 1 - The Main Program

- You should write a program that initialises SLED1 (*You should initially set SLED1 to start at location 0,0 and facing direction 0 degrees (i.e. North)*), opens the two files and then can call the functions you write as your answers in Parts 2 - 3 to test that your program is working satisfactorily. Your `main` function will not be marked - it is just to help you develop your program.

Part 2 - Loading the Map

You should write the following three functions to help SLED1 load and display the map.

- Write a function that initialises SLED1's internal map with the character '**.**' at every location. *10% programming marks.*

```
void initialise_map(char [30][30])
```

-sets the contents of the map array to '.'.

- Write a function that populates the map with the entries in the file. *15% programming marks.*

```
void populate_map(FILE *,char [30][30])
```

-modifies the map array to contain the entries described in the map file.

- Write a function that displays the map, with coordinate 0,0 displayed in the bottom left. Your map should match exactly the example below (for a sample 5x5 map that has been initialised and has a house ('H') at location 0,0), a university ('U') at location 0,4 and a factory ('F') at location 4,4. *15% programming marks.*

```
void print_map(char [30][30])
```

-prints the map array.

```
U...F
.....
.....
.....
H....
```

Part 3 - Driving SLED1

You need to interpret the commands in the command file to move SLED1 around the map.

- For the next function you need to know how to access the fields of a structure when you have a pointer to the structure. This is achieved using the '->' notation. For example, the following code snippet shows how to set the direction a sled is travelling given a pointer to the sled.

```
struct sled my_sled;
struct sled *p_sled = &my_sled;

p_sled->orientation = 90;
```

- You should write a function that takes as parameters a pointer to the command file, the map you read in and a pointer to the structure holding the sled data. The

function should process each command in turn in the file such that when the function finishes all the commands have been processed and the sled structure has been updated appropriately. The function should return 0. *25% programming marks.*

```
int process_command(FILE*, char [30][30], struct sled *)
```

-updates SLED1 to reflect the commands issues.

- Write a function that takes as parameters a structure holding the sled data and outputs the sled data. Your output should match exactly the example below (for a sled that is facing North, is at location X=5, Y=6 and has travelled 25 squares). *5% programming marks.*

```
void print_sled(struct sled)
```

-displays the sled data.

```
sled status:
location 5,6
orientation 0 degrees
distance travelled 25 squares
end sled status.
```

Note, there is a newline at the end of each line of output above - including after ‘`status.`’.*

- Write a function that takes as parameters the map and a structure holding the sled data and outputs the map symbol found at SLED1’s current location. Your output should match exactly the example below (for a sled that is currently at a location that contains a house). *5% programming marks.*

```
void print_symbol(char [30][30], struct sled)
```

-displays the symbol on the map found at SLED1’s current location.

```
sled is flying over symbol H
```

Note, there is a newline at the end of the this line.

3 Week 10’s Practical

In the week 10 practical you will have to modify your program, complete additional tasks worth 25% of the programming marks and then submit the complete program **for plagiarism checking** and test the program using a series of on-line tests during the practical class. In addition you will be asked questions that are designed to

assess your understanding of the program that you have written and of programming in general. The marks will be split 80% programming, 20% questions about the program and programming. *We recommend generating your own test files to ensure your program is well to catch errors in advance of your marking session.*

During the assessment in week 10 you will NOT be able to compare notes with your colleagues.
Nigel Davies and Adrian Friday.