UCP Report

Jakob Wyatt

May 26, 2019

Contents

1 Purpose of Functions

1.1 resolveAdventure

```
status resolveAdventure(map items, unsigned long rows, unsigned
    ↪ long cols, list movements, explorer* person, FILE* file);
```

Resolve an adventure, given a map and a list of movements.
`items` Map containing all items.
`rows` Number of rows in the map.
`cols` Number of columns in the map.
`movements` Movements to be made by the explorer.
`person` Explorer to collect items.
`file` File to write logs to.
Return: SUCCESS if the adventure was sucessful, CORRECTED if out of bounds movements were corrected, FAILED if out of bounds movements weren't corrected.
Precondition: file must be opened in "a" mode.
Postcondition: items is modified when treasure is picked up by person.
Postcondition: items and movements are not deallocated by this function. Define AI to correct out of bounds movements. Define LOG to log movements to stdout as well as file.
Implementation: Iterate through the list of movements, and keep track of the current location of person. This is done in 2 steps:

1. Get the next movement from the list, and calculate the final block person will be on after this movement. This is done with endBlock.

2. Travel in the direction of the movement using moveDist until the final block is reached. Treasures are collected and logged with collectAndLog.

This is done until movements is exhausted or an error status is set.

1.2 collectAndLog

```
void collectAndLog(FILE* file, map items, explorer* person, long
    ↪ i, long j);
```

Collects the treasure at the given location. Logs this event to a file.
`file` File to log information to.
`items` Map containing item to collect.
`person` Explorer to collect the item.
`i` y location of the treasure to collect.
`j` x location of the treasure to collect.
Precondition: file must be opened in "a" mode.
Postcondition: The treasure in items is either swapped, destroyed (and deallocated), or retained.

Implementation: Switch on the type of the treasure. coin: Deallocate and increase explorer.coin by treasure.value. magic: Deallocate and increase explorer.magic by treasure.value. gear: Use treasure.compare to compare and conditionally swap the two treasures. Log with logTreasure if any deallocation or swapping occured.

1.3 logTreasure

```
void logTreasure(FILE* file, treasure x, int collect, long i,
    ↪ long j);
```

Logs a treasure to a file, and optionally stdout.
`file` File to log information to.
`x` Treasure to log.
`collect` If collect == 0, then log x as collected. Else, log it as discarded.
`i` y location of the treasure.
`j` x location of the treasure.
Precondition: file must be opened in ”a” mode. Define LOG to write the information to stdout as well as file.

Implementation: Switch on x.type, and log the treasure according to the assignment specification.

1.4 endBlock

```
status endBlock(unsigned long rows, unsigned long cols, long* i,
    ↪ long* j, move x);
```

Finds the final block after a movement is performed.
`rows` The number of rows in the map.
`cols` The number of cols in the map.
`i` The starting y location. Exports the new y location.
`j` The starting x location. Exports the new x location.
`x` The movement direction and distance.
Return: SUCCESS if the movement was performed correctly, CORRECTED if the movement was corrected, FAILED if the movement is out of bounds. Define AI to correct out of bounds movements.

Implementation: Move the given distance with move_dist. If AI is defined, correct the location if out of bounds. If the distance is still out of bounds, return FAILED.

1.5 move_dist

```
void move_dist(direction dir, unsigned long distance, long* i,
    ↪ long* j);
```

Moves a given distance in a given direction.
`dir` The direction to move.
`distance` The distance to move.
`i` The starting y location. Exports the new y location.
`j` The starting x location. Exports the new x location.
Postcondition: Does not check if the distance is valid.

Implementation: Depending on the direction, change the relevant coordinate.

1.6 insert

```
void insert(list* x, node* iter, void* data);
```

Insert an element into a linked list.
`x` The list to insert into.
`iter` Insert before this element. If iter is NULL, insert at the end of the list.
`data` The pointer to data that will be stored by the node.

Implementation: A node named new is dynamically allocated (on the heap), and data is referenced by it.

- If iter == NULL, insert at the end of the list.
    1. Node before new is x→tail.
    2. Node after new is NULL.
    3. x→tail is now new.
    4. If there are currently no elements in the list, set x→head to new. Otherwise, link the node before new to new.
- Otherwise, the next node is guarenteed to exist.
    1. Node before new is iter→prev.
    2. Node after new is iter.
    3. iter→prev is new.
    4. If iter→prev is NULL (start of list), x→head is new. Otherwise, new→prev→next is new.

1.7 remove_node

```
void remove_node(list* x, node* iter);
```

Remove a node from a linked list.
`x` The list to remove the node from.
`iter` The node to remove. Calls free() on iter.data

Implementation: First, free iter→data. If iter is the first element in x, change the head pointer to iter→next. Otherwise, the previous node.next is equal to the next node. When iter is the final element in x, perform the algorithm above but set the tail pointer and the next node.prev instead.

1.8 make_list

```
list make_list();
```

Initializes an empty list.
Return: The empty list. The list struct is statically allocated (on the stack).

Implementation: Set head and tail to NULL.

1.9 free_list

```
void free_list(list* x);
```

Free all elements in the list.
`x` The list to free. Calls free() on node.data for all nodes.

Implementation: Call remove_node on x→head until x→head is NULL (list is empty).

1.10 for_each

```
void for_each(list x, data_func func);
```

Applies a function to every node in the list.
`x` The list to apply func to.
`func` The function to apply to every node. func is called on each node in order.

Implementation:

1. Apply func to the current node, starting at x.head.
2. Move the current node forward.

Repeat this until the current node is NULL.

2 Conversion of Input File to Coordinate System

2.1 Implementation

The top level function to convert the input file is read_map. This function opens the file, reads and validates the map size (rows and columns), and uses allocate_map to allocate memory for the map.

Once this is done, fill_map is called. This function parses the file line by line, and reads values into the map. First, read_line is called. This function reads a line of arbitrary size by reallocating larger buffers until the line is read in full. This means that the map can be as large as the user wants. Once the line is read, it is split into tokens using split. This function iterates over the provided string, finds a given delimiter (',' in this case) with strchr, and substitutes it will a null-terminator. It then exports an array filled with pointers to the start of each token. This has the effect of breaking a string up into many substrings, seperated by the delimiter.

Once this is done, each individual token is passed to make_treasure. This function uses a switch statement on the first character to determine the type of the treasure. Once the type is determined, the treasure is usually parsed by finding the colon character(s) with strchr, and either using strncpy or sscanf to parse each section. In the case of a gear treasure, the function chooseCompareFunc is used to determine which function pointer should be used with the given slot.

2.2 Alternate Implementation

An alternate approach would be to either use strtok, or the given tokenizer, to parse each line. This would mean that make_treasure would be passed tokens one at a time, rather than splitting the entire string and then passing all tokens to make_treasure. This method was not chosen, as strtok ignores repeated characters and is therefore unsuitable for any empty sections of the map. strtok could also be used to parse individual treasures, instead of using strchr.

3 Sample Input and Output

All test results are generated from the TreasureHunter binary. This is called with the arguments:

```
./TreasureHunter <map file> <list file>
```

A script to run these tests is located in test/all_tests.sh. There is no user input for any test cases. If example output from adventure.log is not provided, it should be assumed that the binary does not provide log file output for that test case.

The TreasureHunterLog binary will produce the same output as the TreasureHunter binary, except any output to the log file (adventure.log) is also output to stdout. This is done real time; as lines are printed to the log file, the same content is printed to stdout.

The TreasureHunterAI binary is assumed to produce the same output as the TreasureHunter binary, unless otherwise stated.

3.1 Example Input from Assignment Brief

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stdout

```
STATUS: COMPLETE
COINS: 50
MAGIC: 445
GEAR: 990
```

Log file

```
---
COLLECT<ITEM:MAGIC, XLOC:2, YLOC:0, DESCRIPTION:Healing Potion,
    ↪ VALUE:85>
COLLECT<ITEM:MAGIC, XLOC:2, YLOC:2, DESCRIPTION:Defence
    ↪ Enchantment, VALUE:360>
COLLECT<ITEM:GEAR, XLOC:4, YLOC:2, DESCRIPTION:Lightsaber, SLOT:
    ↪ hands, VALUE:850>
COLLECT<ITEM:GEAR, XLOC:1, YLOC:1, DESCRIPTION:Vibranium Shield,
    ↪ SLOT:hands, VALUE:990>
DISCARD<ITEM:GEAR, XLOC:1, YLOC:1, DESCRIPTION:Lightsaber, SLOT:
    ↪ hands, VALUE:850>
COLLECT<ITEM:COINS, XLOC:1, YLOC:3, VALUE:50>
```

3.2 Out of Bounds

3.2.1 TreasureHunter

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
LEFt 3
RIGHT 1
down 2
LEfT 5
RIGHT 20
left 1
DOWN 100
UP 2000
```

stdout

```
STATUS: FAILED
```

Log file

```
---
```

3.2.2 TreasureHunterAI

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
LEFt 3
RIGHT 1
down 2
LEfT 5
RIGHT 20
left 1
DOWN 100
UP 2000
```

stdout

```
STATUS: CORRECTED
COINS: 0
MAGIC: 736
GEAR: 32031
```

Log file

```
---
COLLECT<ITEM:GEAR, XLOC:0, YLOC:1, DESCRIPTION:Thorn armour, SLOT
    ↪ :chest, VALUE:120>
COLLECT<ITEM:GEAR, XLOC:1, YLOC:1, DESCRIPTION:Vibranium Shield,
    ↪ SLOT:hands, VALUE:990>
COLLECT<ITEM:MAGIC, XLOC:2, YLOC:0, DESCRIPTION:Healing Potion,
    ↪ VALUE:85>
COLLECT<ITEM:MAGIC, XLOC:2, YLOC:2, DESCRIPTION:Defence
    ↪ Enchantment, VALUE:360>
COLLECT<ITEM:MAGIC, XLOC:3, YLOC:1, DESCRIPTION:Phoenix Blood,
    ↪ VALUE:291>
COLLECT<ITEM:GEAR, XLOC:4, YLOC:1, DESCRIPTION:IDEK_ANYMORE, SLOT
    ↪ :head, VALUE:21>
COLLECT<ITEM:GEAR, XLOC:5, YLOC:1, DESCRIPTION:Crimson Plate,
    ↪ SLOT:chest, VALUE:1020>
DISCARD<ITEM:GEAR, XLOC:5, YLOC:1, DESCRIPTION:Thorn armour, SLOT
    ↪ :chest, VALUE:120>
COLLECT<ITEM:GEAR, XLOC:6, YLOC:1, DESCRIPTION:Infinity Pants,
    ↪ SLOT:legs, VALUE:30000>
```

3.3 Empty List

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
```

stderr

```
File <list file> was empty.
```

3.4 Invalid Direction

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
LEFT 3
riGhtt 1
DOWN 2
LEFT 5
RIGHT 20
NOrtH 1
DOWN 100
UP 2000
```

stderr

```
Invalid direction
At line 2.
```

3.5 Invalid Distance

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
DOWN 2
RIGHT 2
DOWn %s
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Distance must be an integer
At line 3.
```

3.6 No Space in List

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT1
UP 3
riGHT 0
RIGHT 1
```

stderr

```
No space
At line 5.
```

3.7 Negative Distance

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT -100
```

stderr

```
Distance must be positive
At line 8.
```

3.8 Floating Point Distance

Map

```
5,4
,,C 200,
,G Vibranium Shield:hands:990,,C 50
M Healing Potion:85,,M Defence Enchantment:360,
,,,
,,G Lightsaber:hands:850,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0.5
RIGHT 100
```

stderr

```
Distance must be an integer
At line 7.
```

3.9 Empty Map

Map

```

```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting in <map file>, line 1: Expected positive
    ↪ integers <rows>,<cols>
```

3.10 Incorrect Rows

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect number of rows: read 6, expected 7.
```

3.11 Incorrect Columns

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect number of columns at line 7: expected 3.
```

3.12 Negative Columns

Map

```
7,-3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting in <map file>, line 1: Expected positive
    ↪ integers <rows>,<cols>
```

3.13 Invalid Rows

Map

```
7.5,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting in <map file>, line 1: Expected positive
    ↪ integers <rows>,<cols>
```

3.14 Invalid Treasure Type

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,f Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
F is not a valid treasure type.
At row 5, column 3.
```

3.15 Invalid Slot

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G bow and arrows:feet:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting. Gear is represented as: "G <detail>:<slot
    ↪ >:<value>"
At row 5, column 3.
```

3.16 Invalid Value

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:99.5,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting. Gear is represented as: "G <detail>:<slot
    ↪ >:<value>"
At row 2, column 2.
```

3.17 Too Many Colons

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing:Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting. Magic items are represented as: "M <detail
    ↪ >:<value>"
At row 3, column 1.
```

3.18 Too Little Colons

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
M Healing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting. Gear is represented as: "G <detail>:<slot
    ↪ >:<value>"
At row 7, column 2.
```

3.19 No Space in Map

Map

```
7,3
,g Thorn armour:CheST:120,C 200
,G Vibranium Shield:hands:990,
MHealing Potion:85,,M Defence Enchantment:360
,m Phoenix Blood:291,
,g IDEK_ANYMORE:heaD:21,G Lightsaber:hAnDs:850
c 350,g Crimson Plate:cHeSt:1020,
,G Infinity Pants:leGS:30000,
```

List

```
DOWN 2
RIGHT 2
DOWn 2
up 0
LEFT 1
UP 3
riGHT 0
RIGHT 2
```

stderr

```
Incorrect formatting. Magic items are represented as: "M <detail
    ↪ >:<value>"
At row 3, column 1.
```