PA1: Programming Assignment 1

Topic: path finding (one of the primary areas of game programming)

Due date: 2/5, 9 AM (new time)

To be done individually

No need to turn in a CD-ROM
The Game

Rescue a victim from a building, or defuse a bomb in the building, whichever of the victim or the bomb is closer.

The Building: \( n \) levels, \( 1 \leq n \leq 10 \).

Level 0 is always the basement, level 9 the roof.

Each level is an \( N \times N \) square, \( N \geq 4 \).

Each level can have different dimension and orientation.
Staircases

Each level can be connected to another level by a staircase.

Staircases are bidirectional: if both levels exist, the staircase connecting them must exist on both levels.

It is ok for a level to be inaccessible.

It is ok for a staircase to lead to a non-existent level.

Two levels cannot be directly connected by more than one staircase.
Locations

Starting position: ‘S’

Bomb location: ‘B’

Victim location: ‘V’

Walls: ‘x’

All case sensitive

Staircases: marked by the destination level number
Obstacles

You cannot walk through walls.

Staircases are not considered obstacles.

Some walls may have crumbled, but you can’t step off the $N \times N$ grid of each level.
Input File

version=281.04.i
nlevels=1
level=1
dimension=8
# this is a comment
map=
Vxxxxxxxx
x B
x x xxx
x x x
x xx
x xx
x x
x Sx
xxxx2xxx
Directions

Top of file is north.

Direction markers:
\( n: \) north; \( s: \) south; \( e: \) east; \( w: \) west;
\( h: \) northwestern; \( j: \) southwestern; \( k: \) northeastern; \( l: \) southeastern.

Most northwesterly tile is coordinate \((0,0)\).

Staircases do not have to start and end at the same coordinate at both levels.
Output File

# output for file test.txt
version=281.04.o
nlevels=1
level=1
dimension=8
map=
Vxxxxxxxx
x neee
x xnxxx
x x hx
x xxn
x h x
x Sx
xxxx2xxx
Path Marking

Starting from the tile adjacent to the ‘S’tart position, mark each tile with the direction taken in the previous step.

Overwrite the target tile with the direction of the final step taken.

Upon stepping on a staircase tile, overwrite the staircase tile with the direction of the previous step taken. If the stairs are taken, do not overwrite the staircase tile on the exit level.

When standing on a staircase tile, all adjacent tiles on both levels are visible.
Error and Exit Codes

If you run into an error, print out an error message and call `exit()` with the appropriate error code.

Upon correct exit, also call `exit()` with the appropriate exit code.
Map Representation

Each level can be stored as a multi-dimensional array or a sparse table. Use only one, not both.

A multi-dimensional array must be implemented as a single contiguous chunk of memory (you can call \texttt{new} only once, though you may use other helper structures to access the array).

As you read in the map, remember where the staircases at each level are.

You are \textit{not} to remember where the bomb and victim are.
Path Finding

Implement three algorithms:

1. queue based
2. stack based
3. third algorithm
Queue-based

enqueue start position

loop {
    dequeue next tile
    enqueue all unique adjacent tiles
} until target reached or queue is empty

if (queue is empty) search failed

if (target reached) show path taken

Pick one of multiple possible paths
Stack-based and Third

Stack-based: same as before, except use push and pop instead of enqueue and dequeue

One of queue-or stack-based must find the path with the lowest cost (least number of tiles traversed)

For the third algorithm, you are allowed to remember where the bomb and victim are.

Third algorithm must find lowest cost path faster than the fastest of the above two

Sugih Jamin (jamin@eecs.umich.edu)
Timing Your Code

Measure only the time it takes your path-finding algorithm to run, don’t include time to do I/O. For example:

```c
init_stuff();
read_map();

gettimeofday(&start,0);
path_find();
gettimeofday(&end,0);
timing = end-start; /* PSEUDO-CODE! */

report(timing);
```

Time all three algorithms, for \( M \) different-sized maps, then plot the results using gnuplot.
Command Line Options

- `q`: run queue-based algorithm
- `s`: run stack-based algorithm
- `t`: run third algorithm
- `i <string>`: take string as input file name, default stdin
- `o <string>`: take string as output file name, default stdout
- `e`: run timing analysis, must be used with one of `-q`, `-s`, or `-t`
- `h`: help

- `q`, `-s`, and `-t` are mutually exclusive

How do you read the command line options?
Never seen `main(argc, argv)`
Other Items

Coding style

Empirical efficiency

Testing your code (may post test cases and outputs)

Grade composition

What to turn in

The autograder (may be delayed)