CSC3200 - FALL 2025 - ASSIGNMENT 3

DEADLINE: NOV 3TH 23:59

1. Maze (50%)

Please review the Maze example in lecture 12. For this problem, your objective is to implement the search_maze function based on the logic given in the example. The function prototype is given in the maze.h attached to this assignment. You may use the std::stack data structure from the STL. Every maze cell marked as "free" that is discovered during the search must be marked visited in the passed visited matrix by setting the corresponding entry to true. The search must start at the position indicated by the start argument and end at the position indicated by end, so that no further mace cell is discovered from the end cell. Use the passed stack to maintain the search frontier.

```
using maze_t = std::vector<std::vector<maze_cell>>;
using visited_t = std::vector<std::vector<bool>>;
// ...
void search_maze(maze_t const& maze, maze_pos start, maze_pos end,
                 std::stack<maze_pos>& stack, visited_t& visited);
```

Remark. You may assume that visited and maze are matrices (vectors of vectors) of appropriate size. Every entry in visited is false. You may assume that the passed stack is empty.

Remark. You may use the print and validate_dfs methods for debugging.

Remark. You may use the STL but no external graph libraries or graph algorithms.

2. Priority Queue (50%)

Implement a container adapter class from scratch as a public member of your vector class, i.e.

```
namespace student_std {
template <typename T, typename Container = std::list<T>>
class priority_queue {
public:
    using container_type = Container;
    using value_type = typename Container::value_type;
    using size_type = typename Container::size_type;
// ...
};
}
```

It must support the types shown above, you may include <list> for this. Aside from that:

- It must be default constructible (with a default constructed Container), copy constructible, and assignable.
- It must support T const& top(), void pop(), size_type size(), bool empty(), void push(const T&), and void swap(priority_queue&) with const members where appropriate.
- While Container defaults to std::list<T>, your type must also work if e.g. an std::vector<T> is passed or another container that supports the operations front(), pop_front(), push_front(), swap(Container&), size(), empty(), begin(), end() (returning at least a forward_iterator), and insert(Container::iterator, T const&).
- swap must swap the contants of two priority queues without making copies. You may assume that a container.swap(Container&) member is available in the container class.
- **top() must return the largest element in the queue with a fixed number of operations independent of current size**, where "largest" is determined by the operator<, that you may assume exists for T. **Note, that this requires maintaining a suitable invariance**. push(const T&) is allowed to take more time, proportional to the current length of the queue.

Submission Format

Two files: student_maze.cpp, student_priority_queue.h.