```
queue.enqueue(number);
queue.enqueue(stack.pop());
queue.enqueue(queue.dequeue);
queue.enqueue(queue.dequeue);
```

// content of stack from bottom to top: 3, 2, 1, 4, 5

// content of queue from head to tail: Empty

2. (10 points)

Determine which data structure is most suitable in different scenarios:

a) A web browser remembers the web pages you have visited. This information is used when you go back one page. Note that you can go back one page repeatedly. Which abstract data type is most suitable to store the visited web pages for the operation that allows the user to go back one page, possibly repeatedly? Stack

b) Consider the basic version of the domino game where domino tiles are played in a series of tiles. Adjacent tiles must have matching values. A player can only add tiles at one of the two ends of the line of tiles. What abstract data type would be most suitable to store the played tiles? List

3. (10 points) 5

Write pseudocode that reverses the order of the elements in a queue using one additional queue or one stack, and possibly some variables. Do not use any other additional collection. The only valid stack operations are isEmpty, push, pop, and peek. The only queue operations are isEmpty, enqueue, dequeue, and peek.

```
Queue with data
Stack stack = new Stack;
while(queue is not Empty){
    stack.add(queue.peek);
    queue.remove();
3 }
while(Stack is not Empty){
    queue.add(Stack.peek);
    Stack.pop();
3 }
```

4. (10 points) C

Write pseudocode that reverses the order of the elements in a stack using one additional stack and possibly some variables. Do not use any other additional collection. The only valid stack operations are isEmpty, push, pop, and peek. You are not allowed to use a pointer or reference for a stack, that is, an operation of the form "stack1 = stack2;" is not valid where stack1 and stack2 are stacks.

```
stack1 existing
Stack 2 is new
while(stack1 is not Empty){
    stack2.push(stack1.pop());
}
return stack2
```

5. (10 points)

Below is a maze where each cell is labeled by a letter. Each cell can be modeled as a node. The goal is to determine a path from the source node A to the destination node Y. Apply DFS by using the generic search algorithm where the collection `nextNodes` is implemented as a stack. The neighbors of a cell are traversed clockwise starting at the upper cell. For example, if the neighbors of cell H are added to the stack in case of DFS, they are added in the order C, I, G. In particular, G will be at the top of the stack and therefore the search will continue removing G from nextNodes in the next iteration.