ADT — description of behavior that a data structure can exhibit
List

Data Structure — collection of algorithms & data defns which provide ADT's behavior
Array List
Linked List

Solve maze:

- Where have we been?
- Where am I?
- What are my choices?
- Where can I go eventually? Where haven't I been?

Where have I been?	Where haven't I been?
front walk	flower garden
front of garage	sitting room
porch	dining room
foyer	

Depth-First Search

DFS

ADT: Stack

void	push(T)	—	add an element
T	pop()	—	remove the most recently added but not yet removed element
T	peek()	—	discover what pop would return
int	getSize()	—	how many elements?

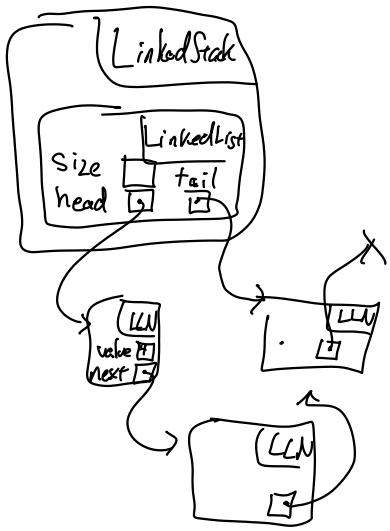
Data Structures

Linked Stack

push	insertFront	O(1)
pop	removeFront	O(1)
peek	getFront	O(1)
getSize	list.getSize	O(1)

ArrayStack

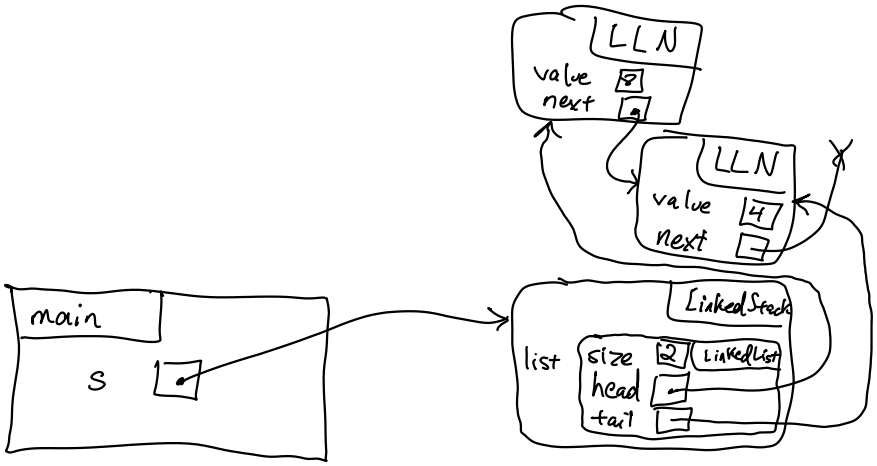
insertBack	O(1) amortized
removeBack	O(1)



Stack Diagrams & Static Allocation

```
Stack<int>* s = new LinkedStack<int>();
s->push(4);
s->push(8);
cout << s->pop() << endl;
```

```
class LinkedStack : public ... {
  private:
    LinkedList<T> list;
    ⋮
};
```



Object Lifecycle

1. Allocation
   a. Find memory
   b. Static initialization — run constructor for all statically allocated objects
   c. Constructor
2. Use
3. Deallocation
   a. Destructor
   b. Static uninitialization — run destructor ...
   c. Release memory