Structures

Structures

- A data structure is a group of elements grouped together under one name
- The elements, known as members, can have different types

```
struct type_name {
        member_type1 member_name1;
        member_type2 member_name2;
        member_type3 member_name3;
};
```

Example

```
struct location
{
        int x;
        int y;
} ;

location a;
a.x = 0;
a.y = 5;
```

Example: Contact List

```
struct contact{
        string name;
        int phoneNumber;
        string email;
        string group;
} //contact is like a type (e.g. int, double, etc.)
//use it to declare an array of type contact
vector<contact> contactList;
contact[100] contactList;
```

Practice

- NASCAR wants to rank its drivers by points.
- Getting number of drivers from the user
- Each driver will have the format of
  - LastName Number Points
- Output all drivers sorted by points from greatest to least

Input:
```
3
Patrick 10 5
Gordon 24 3
Earnhardt 88 8
```

Output:
```
1. Earnhardt 88 8
2. Patrick 10 5
3. Gordon 24 3
```

Harder Practice

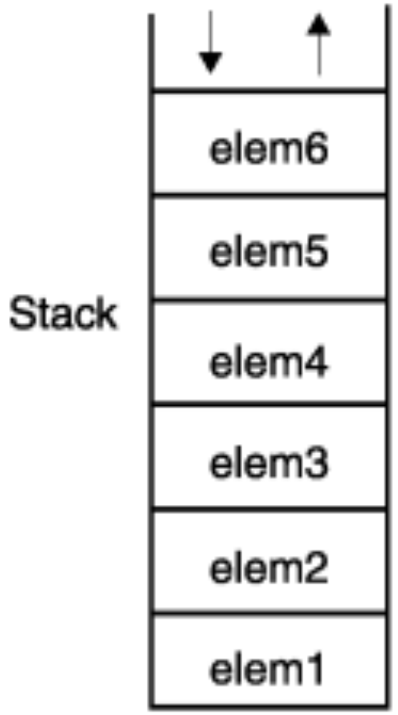
- http://wcipeg.com/problem/ccc11j4

Stack

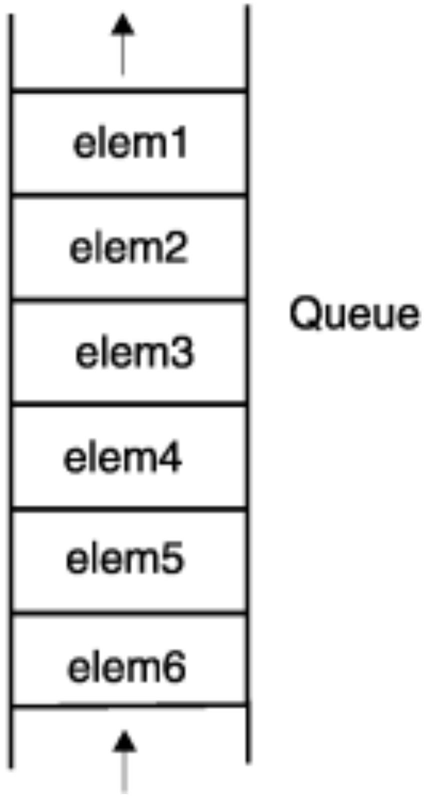
- Last In First Out (LIFO) data structure
- Elements can only be inserted and extracted from one end
- Key elements of a stack
  - empty(), checks if a stack is empty
  - size(), gets size of a stack
  - top(), gets top element of stack
  - pop(), removes top element
  - push(), inserts element into top

Stack Vs. Queue

- LIFO (Last in first out)



FIFO (First in first out)



Code

```
#include <stack>
stack<int> s;
s.push(0);
s.push(10);
s.push(100);
cout << s.top() << endl;
s.pop();
cout << s.top() << endl;
```

```
#include <queue>
queue<int> s;
s.push_back(0);
s.push_back(10);
s.push_back(100);
cout << s.front()<< endl;
s.pop();
cout << s.front()<< endl;
```

Hard Practice

- http://wcipeg.com/problem/ccc14s3