This is CS50.
cs50.brianyu.me
Week 3

- Searching (Linear, Binary)
- Sorting (Bubble, Selection)
- Big O
- Structs
- Recursion
- Merge Sort
What questions do you have?
Today

Sorting

Recursion

Structs
PART ONE

Search and Sort
Bubble Sort
Repeat n-1 times
   For j from 0 to n - 2
     If j'th and j + 1'th elements out of order
        Swap them
wget http://cdn.cs50.net/2020/spring/classes/3/sorting.c
Exercise

Complete `sorting.c` such that it sorts integers using Bubble Sort.

```
wget http://cdn.cs50.net/2020/spring/classes/3/sorting.c
```

Repeat n-1 times
For j from 0 to n - 2
  If j'th and j + 1'th elements out of order
    Swap them
Bubble Sort

for (int i = 0; i < n - 1; i++)
{
    for (int j = 0; j < n - 1; j++)
    {
        if (values[j] > values[j + 1])
        {
            int temp = values[j];
            values[j] = values[j + 1];
            values[j + 1] = temp;
        }
    }
}
Bubble Sort

for (int i = 0; i < n - 1; i++)
{
    for (int j = 0; j < n - 1 - i; j++)
    {
        if (values[j] > values[j + 1])
        {
            int temp = values[j];
            values[j] = values[j + 1];
            values[j + 1] = temp;
        }
    }
}
Bubble Sort

```java
for (int i = 0; i < n - 1; i++)
{
    int swaps = false;
    for (int j = 0; j < n - 1 - i; j++)
    {
        if (values[j] > values[j + 1])
        {
            int temp = values[j];
            values[j] = values[j + 1];
            values[j + 1] = temp;
            swaps = true;
        }
    }
    if (swaps == false)
        break;
}
```
Selection Sort
13482576
For i from 0 to n-1
  Find smallest item between i'th item and last item
  Swap smallest item with i'th item
Exercise

Complete `sorting.c` such that it sorts integers using Selection Sort. wget http://cdn.cs50.net/2020/spring/classes/3/sorting.c

For i from 0 to n-1
  Find smallest item between i'th item and last item
  Swap smallest item with i'th item
Selection Sort

For i from 0 to n-1
Find smallest item between i'th and last item
Swap smallest item with i'th item

for (int i = 0; i < n - 1; i++)
{
    int min_index = i;
    for (int j = i + 1; j < n; j++)
    {
        if (values[j] < values[min_index])
        {
            min_index = j;
        }
    }

    int temp = values[i];
    values[i] = values[min_index];
    values[min_index] = temp;
}
Mergesort
| 3 | 4 | 5 | 8 | 1 | 2 | 6 | 7 |
PART TWO

Recursion
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
void countdown(int n) {
    if (n == 0) {
        return;
        printf("%i\n", n);
        countdown(n - 1);
    }
}
```c
void countdown(int n)
{
    if (n == 0) return; // base case
    printf("%i\n", n);
    countdown(n - 1);
}
```
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1); // recursive call
}
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
Exercise

Write a program `factorial.c` that calculates the factorial of a number.
e.g. Factorial of 5 is $5 \times 4 \times 3 \times 2 \times 1 = 120$

$ ./factorial$

Number: 5

Factorial is 120.
Exercise

Write a program `fib.c` that prints the nth Fibonacci number. 0th is 0, 1st is 1, all others are sum of two previous.

$ ./fib

Number: 5

3
PART THREE

Structs
typedef struct
{
    string name;
    string number;
}
person;
typedef struct
{
    string name;
    string number;
}
person;

person p;
p.name = "Emma";
p.number = "555-0100";
Problem Set 3
Problem Set 3

• Plurality
• One of:
  • Runoff
  • Tideman
VOTE

☐ Alice
☐ Bob
☑ Charlie
VOTE

☐ Alice
☐ Bob
☑ Charlie
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
</table>

**Votes:**

<table>
<thead>
<tr>
<th>VOTE</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>✔️ Alice ✔️ Charlie</td>
<td>✘ Alice</td>
<td>✔️ Bob ✔️ Charlie</td>
</tr>
<tr>
<td></td>
<td>✘ Bob</td>
<td>✔️ Charlie</td>
<td>✘ Charlie</td>
</tr>
<tr>
<td></td>
<td>✔️ Alice</td>
<td>✘ Bob</td>
<td>✔️ Charlie</td>
</tr>
<tr>
<td></td>
<td>✘ Alice</td>
<td>✔️ Bob</td>
<td>✔️ Charlie</td>
</tr>
<tr>
<td></td>
<td>✔️ Alice</td>
<td>✘ Bob</td>
<td>✘ Charlie</td>
</tr>
<tr>
<td></td>
<td>✘ Alice</td>
<td>✔️ Bob</td>
<td>✘ Charlie</td>
</tr>
<tr>
<td></td>
<td>✔️ Alice</td>
<td>✘ Bob</td>
<td>✘ Charlie</td>
</tr>
<tr>
<td>name</td>
<td>Alice</td>
<td>Bob</td>
<td>Charlie</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>votes</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VOTE

- [x] Alice
- [ ] Bob
- [x] Charlie

VOTE

- [x] Alice
- [ ] Bob
- [x] Charlie

VOTE

- [x] Alice
- [ ] Bob
- [x] Charlie

VOTE

- [x] Alice
- [ ] Bob
- [x] Charlie

VOTE

- [x] Alice
- [x] Bob
- [ ] Charlie

VOTE

- [x] Alice
- [ ] Bob
- [x] Charlie
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Alice</td>
<td>Bob</td>
<td>Charlie</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>name</td>
<td>Alice</td>
<td>Bob</td>
<td>Charlie</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

VOTE

- Alice
- Bob
- Charlie

VOTE

- Alice
- Bob
- Charlie

VOTE

- Alice
- Bob
- Charlie

VOTE

- Alice
- Bob
- Charlie

VOTE

- Alice
- Bob
- Charlie

VOTE

- Alice
- Bob
- Charlie

VOTE

- Alice
- Bob
- Charlie
Plurality Vote

• Every voter chooses one candidate.

• Whichever candidate has the most votes wins.
$ ./plurality Alice Bob Charlie
Number of voters: 4
Vote: Alice
Vote: Bob
Vote: Charlie
Vote: Charlie
Charlie
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
typedef struct
{
    string name;
    int votes;
} candidate;

candidate candidates[MAX];
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

candidate candidates[MAX];
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

candidates[0]
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

candidates[1]
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

candidates[2]
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

candidates[2].name
<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

`candidates[2].votes`
Ranked-Preference Voting
VOTE

1. ________
2. ________
3. ________

Alice
Charlie
Bob
Runoff Vote

- Every voter ranks their preferences.
- If a candidate has a majority (more than half) of the votes, they are the winner.
- Otherwise, eliminate the candidate with the fewest votes and re-run the election without them.
7 votes.

4 votes to win.
Alice

Bob

Charlie

VOTE

1. Alice
2. Bob
3. Charlie
Alice

Bob

Charlie
<table>
<thead>
<tr>
<th></th>
<th>Vote 1</th>
<th>Vote 2</th>
<th>Vote 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>Alice</td>
<td>Alice</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>Bob</td>
<td>Bob</td>
</tr>
<tr>
<td>3</td>
<td>Charlie</td>
<td>Charlie</td>
<td>Charlie</td>
</tr>
</tbody>
</table>

Alice

1. Alice
2. Bob
3. Charlie

Bob

1. Bob
2. Alice
3. Charlie

Charlie

1. Bob
2. Charlie
3. Alice
VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Charlie
3. Bob

VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

VOTE
1. Bob
2. Charlie
3. Alice
Alice
1. Alice
2. Bob
3. Charlie

Bob
1. Bob
2. Alice
3. Charlie

VOTE

Alice
1. Alice
2. Bob
3. Charlie

Bob
1. Bob
2. Charlie
3. Alice

Charlie
1. Charlie
2. Alice
3. Bob

Alice
1. Alice
2. Bob
3. Charlie

Bob
1. Bob
2. Alice
3. Charlie

Charlie
1. Charlie
2. Alice
3. Bob

VOTE

Alice
1. Alice
2. Bob
3. Charlie

Bob
1. Bob
2. Charlie
3. Alice

Charlie
1. Charlie
2. Alice
3. Bob

Alice
1. Alice
2. Bob
3. Charlie

Bob
1. Bob
2. Alice
3. Charlie

Charlie
1. Charlie
2. Alice
3. Bob

4

3
Representing Candidates

<table>
<thead>
<tr>
<th>name</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>3</td>
<td></td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>2</td>
<td></td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>4</td>
<td></td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>Dave</td>
<td>0</td>
<td></td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>
typedef struct {
    string name;
    int votes;
    bool eliminated;
} candidate;
Representing Candidates

candidate candidates[MAX_CANDIDATES];
Representing Preferences

```c
int preferences[MAX_VOTERS][MAX_CANDIDATES];
```

`preferences[i][j]` is the candidate index of voter `i`'s preference `j`. 
preferences

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

candidates

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>votes</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eliminated</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

preferences[0][0] = 2;

First voter's top preference is Charlie.
preferences

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

candidates

<table>
<thead>
<tr>
<th>name</th>
<th>Alice</th>
<th>Bob</th>
<th>Charlie</th>
</tr>
</thead>
<tbody>
<tr>
<td>votes</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eliminated</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

preferences[0][1] = 0;

First voter's second preference is Alice.
This is CS50.