Today:

Hash Tables:

- Open addressing
  - ADT
  - coding example

- Assignment 6 - 24 hr extension
- Extra credit challenge

Abstract Data Type for Hash Table with open addressing

```
private:
    hash table
    table size
    hashF(key)

public:
    Init()  // constructor
    insert(record)
    search(key)
    delete(key)
    display()
```

```
struct Record {
    string key;
```

delete(key)
display()

string key;
int data
:
}

void insertRecord(Record r)
(open addressing approach)

1) get hash index based on r.key

2) If table[index] is empty
↳ insert new record @ index

else
iterate table until next available slot is found

(w/ array, use circular approach)

Sample Problem:

Create a Hash Table to store records. The record key is an ASCII string 2 letters long, where each letter can be an upper caps letter of the alphabet (A-Z).

```
struct Record{
    string key;
    // other data could go here
};
```

Generate a set of sample records to test your code.

1) Create a insert() function w/ no collision resolution.
   ↳ experiment w/ table size

2) Update insert() to add collision resolution w/ open addressing.

Range of possible values:

min = A + A = 130

max = Z + Z = 90 + 90 = 180

m = 180 - 130 = 50 ←

Extra Credit Challenge

Thursday, March 7, 2019 5:42 PM

CHALLENGE ALERT!!!

Submission Date Time - Tuesday March 12, 11:59PM
Email Subject - CSCI2270_CHALLENGE_1
The subject has to be EXACTLY this...
Send an email TO YOUR TA.
Implementation of the challenge will be like the midterm: FROM SCRATCH.
SUBMIT program as .cpp, explanation as .txt and output as screenshot.
Also Explain the Time Complexity of your algorithm in the explanation.txt
(That means write the O(?) notation).

Challenge Problem:

Write a program to merge two BSTs into one.
Try and make your program efficient.