Week 5 Section

This week's section handout has practice with pointers and linked nodes, as well as more practice with Big O.

---

1. Hogwarts - Pointer trace (not on CodeStepByStep)

Trace through the following function and draw the program's memory at the designated spot. Indicate which variables are on the stack and which are on the heap, and indicate orphaned memory. Indicate with a question mark (?) memory that we don't know the values of.

```
struct Quidditch {
  int quaffle;
  int *snitch;
  int bludger[2];
};

struct Hogwarts {
  int wizard;
  Quidditch harry;
  Quidditch *potter;
};
```

```
void gryffindor() {
  Hogwarts *triwizard = new Hogwarts[3];
  triwizard[1].wizard = 3;
  triwizard[1].potter = NULL;
  triwizard[0] = triwizard[1];
  triwizard[2].potter =
    hufflepuff(triwizard);
  triwizard[2].potter->quaffle = 4;
  //DRAW THE MEMORY AS IT LOOKS HERE
}

Quidditch * hufflepuff(Hogwarts * cedric) {
  Quidditch *seeker = &(cedric->harry);
  seeker->snitch = new int;
  *(seeker->snitch) = 2;
  cedric = new Hogwarts;
  cedric->harry.quaffle = 6;
  cedric->potter = seeker;
  cedric->potter->quaffle = 8;
  cedric->potter->snitch =
     &(cedric->potter->bludger[1]);
  seeker->bludger[0] = 4;
  return seeker;
}
```

Include your drawing on the next page:

Thanks to Marty Stepp, Anton Apostolatos, and other CS106B and X instructors and TAs for contributing problems on this handout.

Stack	Heap

2. Big O (not on Code Step By Step)

For these problems, `vec` has N elements and `database` has M elements. Express your answer in terms of M and N.

Function	Big O

`int overlap(Vector<int> &vec, Set<int> &database) {`
`  int total = 0;`
`  for (int itemOne : vec) {`
`  for (int itemTwo : database) {`
`  if (itemOne == itemTwo) {`
`  total++;`
`  } else {`
`  for (int itemThree : vec) {`
`  cout << "Wut" << total;`
`  }`
`  }`
`  }`
`  }`
`  return total;`
`}`	

`int overlap(Vector<int> &vec, Set<int> &database) {`
`  int total = 0;`
`  for (int item : vec) {`
`  total += database.contains(item);`
`  }`
`  return total;`
`}`	

`int overlap(Vector<int> &vec, Set<int> &database) {`
`  if (vec.isEmpty()) {`
`  return 0;`
`  }`
`  int item = vec[0];`
`  vec.remove(0);`
`  int itemInDB = database.contains(item);`
`  return itemInDB + overlap(vec, database);`
`}`	

3. Linked Nodes (not on Code Step By Step)

Write the code that will produce the given "after" result from the given "before" starting point by modifying links between the nodes shown and/or creating new nodes as needed. There may be more than one way to write the code, but do NOT change any existing node's data field value. If a variable does not appear in the "after" picture, it doesn't matter what value it has after the changes are made.

The relevant struct is:

```
struct ListNode {
 int data; // data stored in this node
 ListNode* next; // a link to the next node in the list
 // Constructs a node with the given data and a NULL next link.
 ListNode(int data) {
   this->data = data;
   this->next = NULL;
 }
 // Constructs a node with the given data and the given next link.
 ListNode(int data, ListNode* next) {
   this->data = data;
   this->next = next;
 }
};
```