FINAL REVIEW

Problem Solving with Computers-I

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

Final Exam!

- Final exam pages:
  - Section A: https://ucsb-cs16.github.io/w20/exam/e03a/
  - Section B: https://ucsb-cs16.github.io/w20/exam/e03b/
- Section A (2:00p lecture): Tuesday 3/17, 4p - 7p
- Section B (3:30p lecture): Thursday 3/19 4p - 7p
- Assigned seating, will be posted on Piazza
- Everything we have covered so far is on the exam
- Duration: **3 hours**
- Closed book: no calculators, no phones, no computers
- Only 1 sheet (***double***-sided is ok) of written notes
  - Must be no bigger than 8.5” x 11”
  - You have to turn it in with the exam

Review

- Coding practice (recursion + linked lists)
- Dynamic memory pitfalls
- Pointers
- Structs and Linked List

Take notes!

Recursion and linked list

- Given a linked list, implement each of the following:
  - Find the min value in the linked list
  - Delete all the nodes in the linked list
  - Delete the value of a single node in a linked list

Pointer pitfalls and memory errors

- **Segmentation faults**: Program crashes because it attempted to access a memory location that either doesn’t exist or doesn’t have permission to access
- Examples of code that results in undefined behavior and potential segmentation fault

```
int arr[] = {50, 60, 70};

for(int i=0; i<=3; i++){
   cout<<arr[i]<<endl;
}
```

```
int x = 10;
int* p;
cout<<*p<<endl;
```

Dynamic memory pitfalls

Memory leaks (tardy free):

Heap memory not deallocated before the end of program

Heap memory that can no longer be accessed

Example

```
void foo(){
     int∗ p = new int;

}
```

Dynamic memory pitfalls

Dangling pointer: Pointer points to a memory location that no longer exists

Which of the following functions returns a dangling pointer?

```
int* f1(int num){
    int* mem1 =new int[num];
    return(mem1);
}
```

```
int* f2(int num){
     int mem2[num];
     return(mem2);
}
```

A. f1
B. f2
C. Both
D. Neither

Pointers

1. What C++ unary operator is the "de-referencing" operator?

2.What C++ unary operator is the "address-of" operator?

3. Declare a variable p to be a pointer to a pointer to a character

4. Draw a pointer diagram to show the evolution of data in memory during the execution of the the following code:

A.

```
int a=6, b=7, *p1=&b, *p2=&a;
p1 = p2;
*p1 = 8;
p2 = &b;
```

Draw pointer diagrams

B.

```
int a=2, b, *p1=&b, *p2=&a, *p3;
p3 = p2;
*p1 = 8;
p2 = p1;
p1 = p3;
*p2 = 4;
```

C.

```
int a=2, b=3, *p1, *p2;
p2 = &a;
p1 = &b;
*p1 = *p1 + *p2;
```

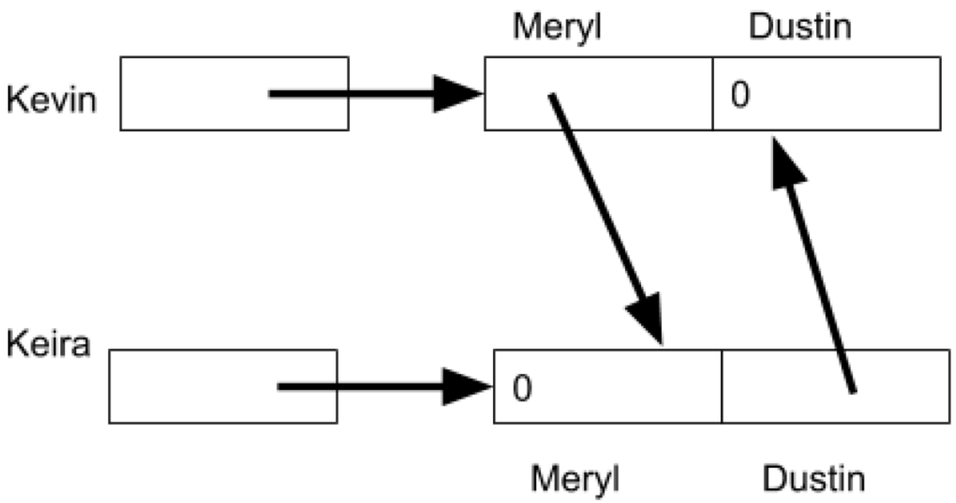
Draw pointer diagrams for the following code

(a) Draw a pointer diagram for the following code:

```
int*** p = new int**;
*p = new int*;
**p = new int;
***p = 5;
```

(b) Write code to print the values of all data created on the heap

Pointers and Structs



Starting with the current state of memory shown above, consider the C++ code shown below. In the space to the right, draw the state of memory after this code executes?

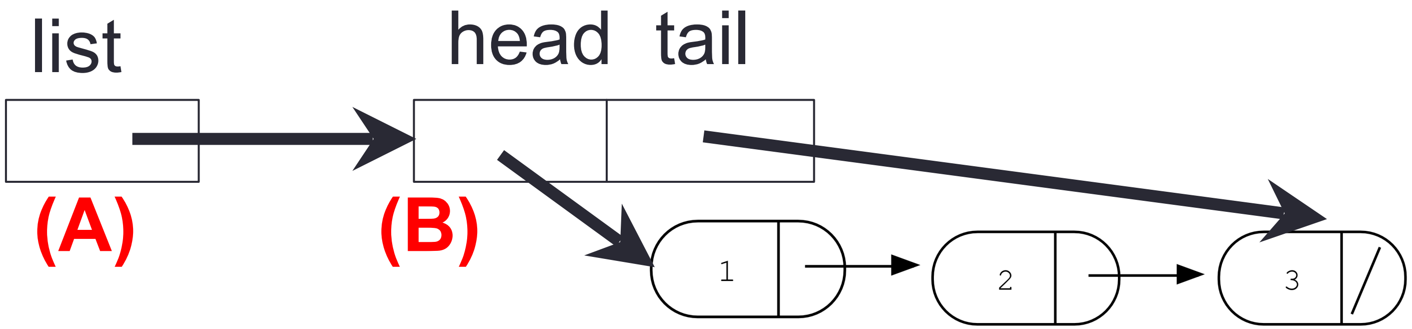
```
struct Actors{
     Actors* Meryl;
     Actors* Dustin;
};
Actors* Kevin;
Actors* Keira;
```

```
Kevin->Meryl = 0;
Kevin->Dustin = Keira;
Keria = Keira->Dustin;
```

Deleting the list

```
int freeLinkedList(LinkedList * list){…}
```

Which data objects are deleted by the statement: `delete list;`



(A)

(B)

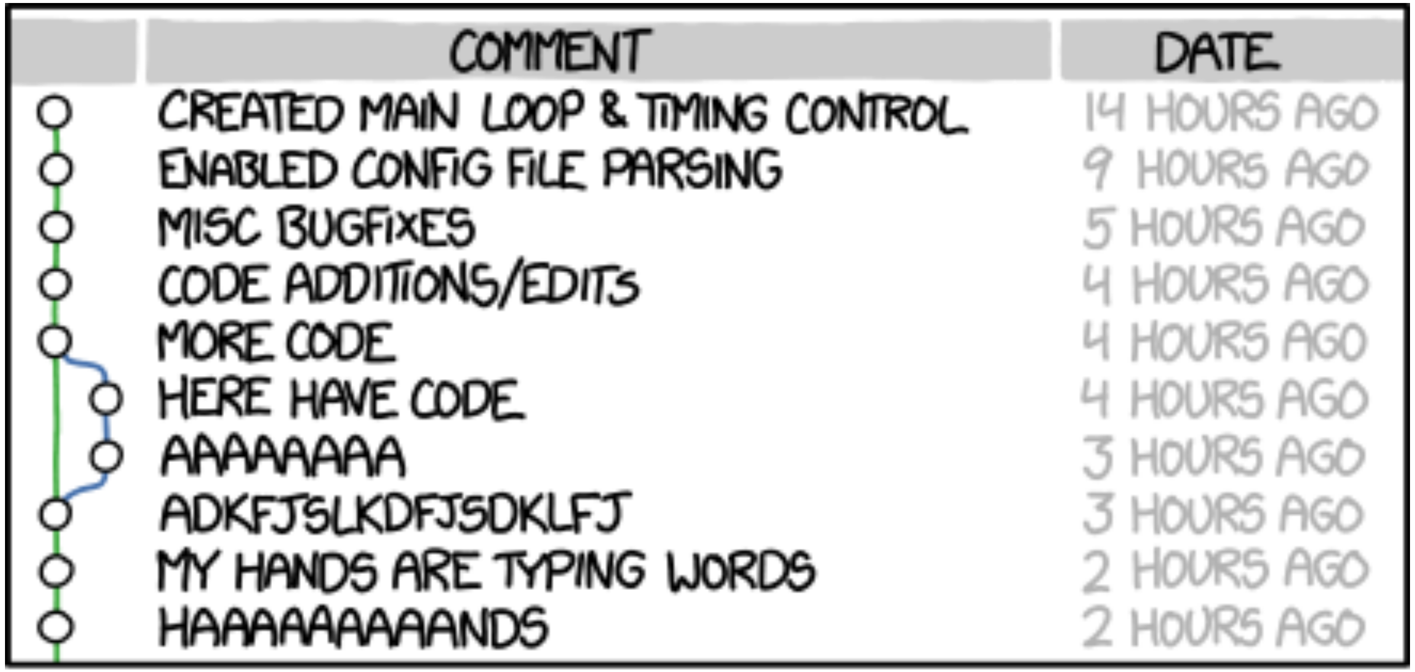
(C) All nodes of the linked list

(D) B and C

(E) All of the above

Does this result in a memory leak?

Some comic relief…



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

HTTP://XKCD.COM/1296/

Some comic relief

HTTP://XKCD.COM/138/

Good luck with the final!