Midterm 1 Review

These questions are designed to help you think about course material. The exam will be mostly short answer or coding.

Research shows that the best way to study for an exam is with other people. In group study, the people who start knowing more learn more (think about why), so don't think that it is only valuable to study with people who know more than you do.

1 Memory Management

1. Fill in the blanks in the following function to correctly de-allocate the memory that was allocated during execution of the function. You may not need to use all of the blanks.

```
int g(int x) {
  int **values = new int*[x-1];
  for (int i = 1; i < x; i++) {

    values[i-1] = new int(i);
  }

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________

  ____________________________________________________________
}
```

2 Pointers

2. Consider the following program. For each print statement there is a blank to the right. Fill in the blank with what the statement would print, or the word "error" if it causes a compile or runtime error.

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

```
class A {
public:
    virtual void printOne(){cout << "A\n";}
    bool equals(A a){return data == a.data;}
    void setData(int input){ data = input;}
    void printTwo(){cout << "AA\n";}
private:
    int data;
};

class B: public A{
public:
    void printOne(){ cout << "B\n";}
    void printTwo(){ cout << "BB\n";}
};

int main(){
    A a;
    B b;
    b.setData(7);
    b.printTwo();                     /*______*/
    A* bPtr = &b;
    bPtr->printOne();                 /*______*/

    bPtr->printTwo();                 /*______*/

    cout << b.equals(*bPtr) << endl; /*______*/
    return 0;
}
```

3. Given the following code, draw a memory diagram showing which variables are allocated on the stack and which are on the heap. Connect any pointers to the memory location that they point to using arrows.

```
int main(void) {
  Point center(0,0);
  Point top(0,1);
  Point right(1,0);
  Point left(-1,0);

  Shape** shapes = new Shape*[3];
  shapes[0] = new Circle(3,center);
  shapes[1] = new Triangle(top, right, center);
  shapes[2] = new Triangle(left, right, top);
  for (int i = 0; i < 3; i++) {
    cout << shapes[i]->area() << endl;
  }

  // clean up memory
  for (int i = 0; i < 3; i++) {
    delete shapes[i];
  }
  delete [] shapes;
}
```

3 C++ Classes

4. Given the following class definitions, draw the a diagram showing the relationship between the classes:

```
class A {
  virtual ˜A();
  // ..
}

class B {
  virtual ˜B();
  // ..
}

struct J {
  A first;
  B second;
  J()
}

class X : public A, public B {
  J data;
  // ..
}
```

4 Algorithm Analysis

5. Use big-O notation to analyze each of the following running times. Make sure to show/explain why your answer is correct.

(a) T(n) = $2n^2 + n + 5$

(b) T(n) = $3 + log(n) + n$

(c) T(n) = $2^n + 5n^3 + 3n^2 + n + 5$

(d) T(n) = $2T(n/2) + O(n)$

(e) T(n) = $6T(n/6) + O(1)$

(f) T(n) = $2T(n/4) + O(n^2)$

- Write a code fragment that has the exact running time described for each of the above T(n) equations.

6. Examine the following function definition.

```
int h(int x){
    if (x < 4) return 1;
    else return h(x-3) * h(x-2) * h(x-1);
}
```

- Draw a tree showing the calls for h(6).
- What is the running time (T(n)) equation for h?
- What is the big-O notation for this running time?

7. Analyze the running time of a function you implemented on HW2.

5 LinkedLists

8. Find the error in the following implementation of an add(int value, int index) function on a singly linked list, and write a correct version:

```
void add(int value, int index) {
  LinkNode *newNode = new LinkNode(value, NULL);
  if (head == NULL || index == 0) {
    newNode->next = head;
    head = newNode;
  }
  else {
    LinkNode *current = head;
    int i = 0;
    while (i++ < index && current->next != NULL) {
      current = current->next;
    }

    current->next = newNode;
  }
}
```

9. Compare a linked list with a vector (resizable array). What operations can be performed faster on a linked list? Give a specific example operation and show its big-O notation for both a linked list and a vector.

10. Complete the following code for implementing a remove(int index) operation for a doubly-linked list. You may not need all the lines.

```
void remove(int index) {
  if (head != NULL) {
    LinkNode *current = head;
    int i = 0;
    while (i++ < index && current->next != NULL) {
      current = current->next;
    }

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________

    ________________________________________________________
  }
}
```

11. Define a structure that contains all of the required data fields for a doubly-linked list node.

6 Stacks/Queues

12. Write a Queue.h file that contains a function definition for each of the three main operations of a Queue (as discussed in class).

13. Compare and contrast a Stack and a Queue. Describe one application (or context where the structure could be used) where you would use each of these structures.

7 Trees

14. Draw the ordered binary tree that results from each sequence of inserts.

    - 10, 12, 4, 3, 5, 7, 15
    - 1, 4, 8, 9, 10, 11
    - 10, 9, 8, 7, 12, 14, 16

15. What is the worst case running time for a binary search on an ordered binary tree? Show an example ordered binary tree and the subsequent binary search that matches this running time.

16. Given the following ordered binary tree, show the steps required to remove(15).

```
      27
     /  \
    15  32
   / \    \
  5  23    45
 /  /
3  18
     \
     20
```