Your name:_____

CS 152
Exam 3
Spring 2018

Data Representation	/ 13
Heaps	/ 25
Graphs	/ 15
Implementation	/ 25
Total	/78

1. No one may leave for any reason and come back to work on his/her test.
   If you need to go to the restroom, go now.

2. You may have 1 sheet of notes, hand-written, double-sided

3. Your backpack must be zipped up.

4. You may not use any electronic devices of any kind. Make sure your cell phone is turned off so it does not ring during the test.

5. You may not wear hats, hoods, sunglasses, or do anything that would obscure your eyes.

6. You may not sit near your partner if you are partnered in programming or lab (within 3 seats in any direction)

8. We will not answer questions during the exam.

9. Do not hold your test up. It must stay on your desk.

10. If you seem to be looking around, you will be moved to the front to allow you to look around without having other students' test in your field of vision.

1. Data Representation (13 pts)

a) (5 pts) Each row represents the same number expressed in different formats. Fill in the blank spots.

Decimal	Unsigned binary (16 bits)	Signed 2's comp (16 bits)	Hex (4 digs)	Octal (6 digs)
85		X		
-85	X		X	X
	X	X	0x0043	X

Bitwise arithmetic – ***remember that bits are numbered from right to left, and numbering starts with 0.***
You are implementing a toy ghost whose eyes glow when the motion sensor detects motion.

b) (4 pts) You first need to detect whether or not the motion sensor is detecting motion. There is a pointer, sensor, that points to a spot in memory where inputs from the motion sensor are placed in bit 0. If bit 0 has the value 1, then motion has been detected. Write the if statement that detects motion.

c) (4 pts) Then you need to turn on the two red lights in the eyes of the ghost. There is a pointer, lights, that points to a spot in memory where outputs to the lights are placed. The two lights are bits 3 and 5. Write the code that writes 1 to those bits to turn the lights on.

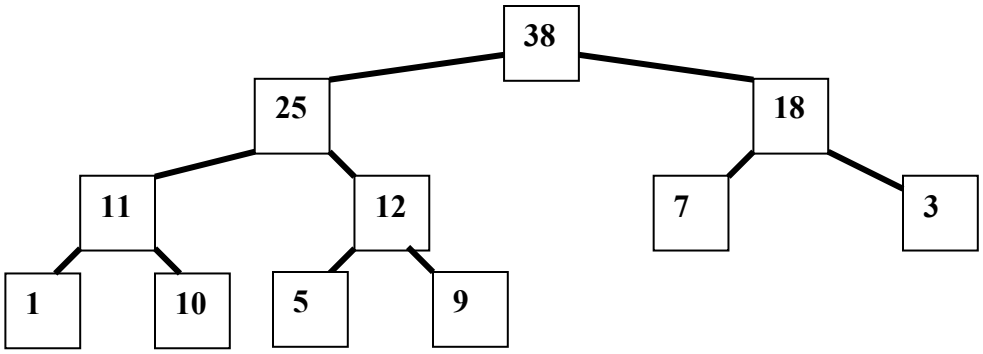
```
 // if motion is detected
if (*sensor                              )


{
        // turn on the two lights
        *lights =


}
```

2. Heaps: (25 pts)

Max-Heap



(5 pts/each) Begin with the max-heap above. Draw the heap after each operation. Do not just draw any heap – it must be the heap that results from the operation on the heap above. **For each operation, begin with the starting heap above – do not use the result from the prior operation.**

a) Enqueue(35)

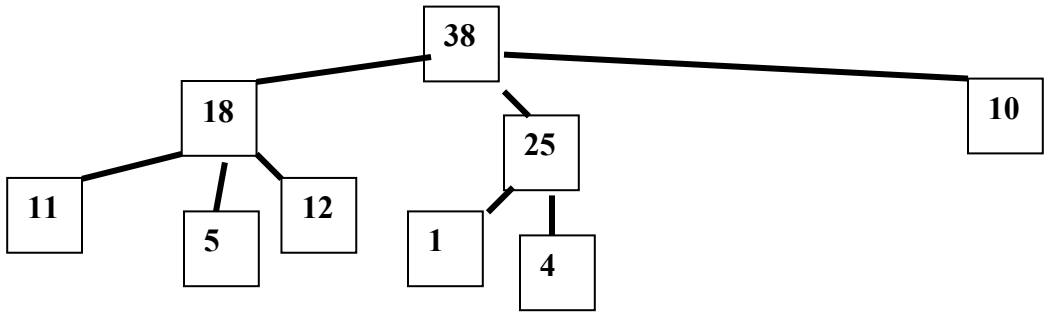
b) Dequeue()

c) (5 pts) One algorithm for sorting is the following: Calculate the computational complexity.

```
// insert each of n items into a heap
for(i=0;i<n;i++)
        heap_enqueue(heap, array[i]);
// remove each of n items from a heap
for(i=0;i<n;i++)
        array[i] = heap_dequeue(heap);
```

Explain your answer.

Heaps (continued)



(5 pts each) Imagine that the heap is stored as a ternary tree in which every node has up to three children. **For each operation, begin with the starting heap above – do not use the result from the prior operation.**

d) Enqueue(40)

e) Dequene()

2. another problem (15 pts)

(Last quarter's problem covered material we did not get to this quarter)

4. Static variables implementation (25 pts)

Static variables were an integral part of iterators. Now we will use them for a different purpose. You are writing a calculator. The calculator can perform several actions, and each action has a result. The result is based on the previous result and the action. Here is code that illustrates how it works (and what the result would be). If someone enters an invalid operation, make no change to the calculated value and return the current value.

```
enum Operation { SET, ADD, SUBTRACT, MULTIPLY, DIVIDE };
int calculate(enum Operation op, int rvalue);

int main()
{
        calculate(SET, 3);              // returns 3
        calculate(SUBTRACT, 3);         // returns 0
        calculate(ADD, 1);              // returns 1
        calculate(ADD, 3);              // returns 4
        calculate(ADD, 3);              // returns 7
        calculate(SUBTRACT, 2);         // returns 5
        calculate(MULTIPLY, 2);         // returns 10
        calculate(DIVIDE, -5);          // returns -2
}

int calculate(enum Operation op, int rvalue)
{









}
```