Binary, hexadecimal, decimal

1. What is the value of

01010101

in decimal? Hexadecimal?
How did you calculate those values?

2. What is 54 in binary, hexadecimal?

3. What is: 110101010111101011101010

in hexadecimal?

Operations counting
4. Given the following Insertion_Sort code, and the unsorted array arr, how many times will the code on lines 40 and 41 execute when j = 2.

```
Arr = [3,4,2,99,1]
```

```
33  void insertionSort(int arr[], int length) {
34      int i;
35      int key;
36      for (int j = 1; j < length; j++) {
37          key = arr[j];
38          i = j-1;
39          while (i >= 0 && arr[i] > key) {
40              arr[i+1] = arr[i];
41              i--;
42          }//end of while loop
43          arr[i+1] = key;
44
45          printArray(arr,5);
46      }//end of for loop
47  }//end of insertionSort.
48
```

How did you calculate your answer?

Pointers
5. After this code executes, what is the value of *b?
int a = 6;
int *b=&a;
int *c=b;
*b=5;
*c=8;
a=12;

How did you determine your answer?

Array operations

6. In this array:

9	12	13	99	123	2	7	90	56	34

how many operations on the array need to be done to produce

9	12	13	99	2	7	90	56	34	

Assume that indexing goes from left to right, starting at 0. An operation is counted as removing or changing a value for any item in the array.

7. In a double-linked list with the same contents, how many operations would it take to remove the item and reset the pointers in the linked list as necessary. You can assume the item has already been found in the list.

8. If you had an array of 100 elements, describe in words how you would add an item in the 50^{th} position. Describe in words how you would remove the 50^{th} item.

Linked lists

7. Assume you are given the following struct:

```
Struct City{
String name;
City *next;
City *previous;
}
```

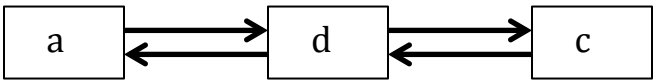
Write the code to generate a double-linked list of cities. Your code should ask the user if they want to add a city. If the user says “yes”, ask for the name of the city. Create the city and add it to the end of the list. Continue asking the user and adding new cities to the end of the list until the user says “no”.

8. Write the code to iterate from the head of a linked list to the tail.

9. Write a code that iterates from the beginning of an array to its end (array size is int arraySize). These iterators should print the value at each element.

10. What are the general steps for adding/removing a node from a linked list? I.e. we have nodes a,b,c in a doubly linked list and we want to remove node b.

11. Assume you are given a linked list that looks like this:



What does the list look like after the following operations are performed:

```
d->setNext(b);
b->setPrevious(d);
c->setPrev(b);
b->setNext(c);
```

12. Using the struct from question 7

If we have the following:

```
City *currNode = new City;
City *next;
```

What happens if we do:

```
currNode->setNext(next);
```

13.
Which of the following code fragments, POP1 or POP2 produces the desired result of popping an item from a single linked list implementation of a stack.

POP1:

```
X = head
head = head->getNext()
Return X
```

POP2:

```
head = head->getNext()
X = head
Return X
```

14. Assume a single-linked list of cities has been created, and you want to iterate through the list and print the name of each city. The following code almost accomplishes this task. What's wrong with this code?

```
City* current = head;
While(current!=NULL){
      Cout<<current->getValue()<<endl;
}
```