You may use any library functions for all questions unless explicitly stated otherwise. If you don't remember the exact call just make sure it is clear what each parameter is and what you expect the function to be doing. For example, if you are looking for a substring, but can't remember the function strstr, you can just write substring, as long as it is clear what you expect it to be doing.

1. *(5 pts)* **Write a C function that takes a *Doubly* linked list as a parameter and reverses the order of the original list. It should not return any value.**

```
void reverse(List * l){
        Node * current = l->head;
        while(current != NULL){
                Node * temp = current->next;
                current->next = current->prev;
                current->prev = temp;
                current = current->next;
        }
}
```

2. *(5 pts)* **Write a single C function that takes a *Singly* linked list as a parameter and reverses the order of the original list. You should use an secondary data structure to accomplish this. It should not return any value.**

```
void reverse(List * l){
        Stack * s = newStack();
        while(l->head != NULL){
                push(s, current->head->data);
                remove(l, 0);
        }

        while(!empty(s)){
                Data d = pop(s);
                insert(l, d);
        }
}
```

For the below pseudo code, abstract out any external functions needed. For example, printNode(node) to print a node's value.

3a) *(3 pts)* **Write pseudocode for a single recursive function that takes a BST root node as a parameter and performs a deep copy of a Binary Search Tree, returning a pointer to the copy BST:**

```
void clone(Node *node, Tree * copy){
    if(node!=NULL){
        insert(copy, node->data);
        clone(node->left, copy);
        clone(node->right, copy);
    }
    return;
}
```

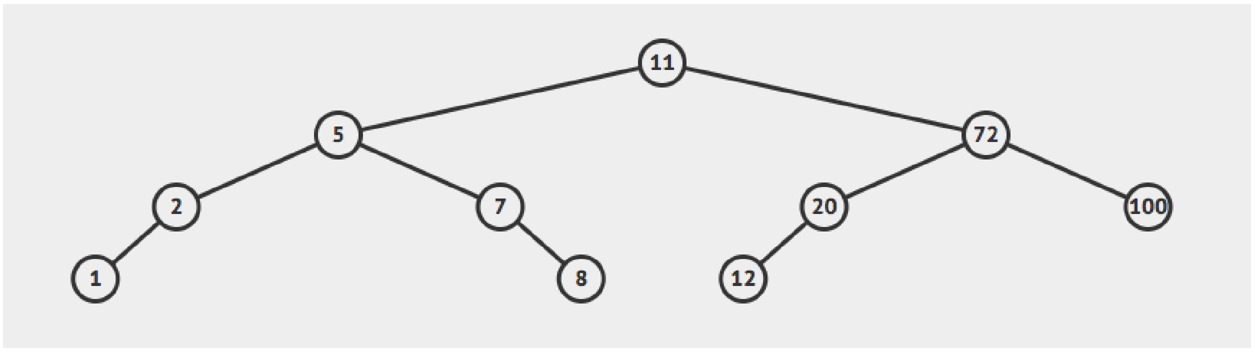
3b) *(3 pts)* **Write pseudocode for a single recursive function that takes a BST root node as a parameter and deletes all nodes in a Binary Search tree:**

```
void deleteTree(Tree * bst, Node *node){
    if(node!=NULL){
        deleteTree(bst, node->left);
        deleteTree(bst, node->right);
        removeLeaf(bst, node);
    }
}
```

3c) *(3 pts)* **Write pseudocode for a single recursive function that takes a BST root node as a parameter and performs a sorted print in a Binary Search tree:**

```
void print(Node *node){
    if(node!=NULL){
        printf("%d", node->data);
        clone(node->left);
        clone(node->right);
    }
    return;
}
```

4. *(5 pts)* **Draw a binary search tree with the values inserted in the following order: 11, 72, 5, 20, 12, 7, 2, 1, 8, 100**



a) How many leaves does the tree have? 4

b) What is the root? 11

c) What is the tree's max height? 3

5. *(5 pts)* **How would the previous Binary Tree appear in memory as an array?**

[11, 5, 72, 2, 7, 20, 100, 1, nil, nil, 8, 12]

a) Show the steps involved when you min-heapify the array

[11, 5, 72, 2, 7, 20, 100, 1, 8, 12]
[11, 5, 72, 1, 7, 20, 100, 2, 8, 12]
[11, 1, 20, 2, 7, 72, 100, 5, 8, 12]
[1, 2, 20, 5, 7, 72, 100, 11, 8, 12]

b) Show the steps involved (both array and logical tree) when you remove the priority value

[12, 2, 20, 5, 7, 72, 100, 11, 8]
[2, 12, 20, 5, 7, 72, 100, 11, 8]
[2, 5, 20, 12, 7, 72, 100, 11, 8]
[2, 5, 20, 8, 7, 72, 100, 11, 12]

IMPORTANT NOTE: The next problem is wildly underspecified and there are many ways to solve it, depending on what data structures you choose to store and what functions you offer to the client. **You must state any assumptions you make and clearly specify any ADT's you use in your solution.** You should implement each struct you will need, but only need the function interfaces.

8. *[10 pts]* Most of you have presumably used a digital music application such as iTunes, probably within the last 24 hours. Such systems allow you to store a database consisting of a number of albums, each of which contains a number of tracks. Your job in this problem is to design the data structure for a new music application called MyTunes that does much the same thing. In MyTunes, the data structure for each track must include the following information:
    a. The title of the song recorded on that track
    b. The artist who recorded it
    c. The running time in seconds
    d. Each album then consists of a number of tracks, along with the title of the album. The MyTunes database records information for as many albums as you choose to store in it.
    e. In this problem, your mission is to define three classes — MTDataBase, MTAlbum, and MTTrack — Code the interface for each of the three classes.
    f. Lastly, state which of the CRUD operations you think is most performance critical for this software and how it influenced your design.