Exam 2 Review

Review Problems Covered in Workshop + More

Stacks

1. **Linked List Implementations**
    a. push()
    b. pop()
    c. top()

2. **Array-Based Implementations**
    a. push()
    b. pop()
    c. top()

3. **Infix to Postfix (Implementation & Tracing)**
4. **Postfix to Infix (Implementation & Tracing)**
5. **Evaluate Postfix (Implementation & Tracing)**

6. **Given an expression, find out whether or not that expression contains valid parentheses. An expression contains valid parenthesis if it has the proper parenthesis ( ), [ ], { } in the correct order.**

string = "{ [ ] (( )) }" → true
string = "{ ( ]) } }" → false

```
bool validParantheses(string s){

}
```

7. **Reverse a stack using recursion (you are only allowed access to the stack passed in through your function, no additional data structures allowed)**

```
void reverseStack(stack<int>& s){

}
```

8. **Given a value, return the index of where it is in the stack (assume the top of the stack is index 0, and the bottom of the stack is index n - 1, n being the size of the stack). If the value doesn't exist in the stack, return -1 (iteratively and recursively)**

```
int returnIndex(stack<int> &s){

}
```

Queues

1. **Linked List Implementations**
    a. enqueue()
    b. dequeue()
    c. front()

2. **Array-Based Implementations**
    a. enqueue()
    b. dequeue()
    c. front()

3. **Priority Queues**
    a. enqueue()

4. **What are the differences between a queue and priority queue?**

5. **Implement a function that takes in two arrays, one that holds a list of tasks, and another that holds each task's associated time to complete and performs round-robin scheduling on the list of tasks (quantum time is the amount of time to perform a task for before moving on to the next). Output the tasks in the order that they're completed.**

```
void roundRobin (string tasksArr[], int taskTimesArr[], int quantumTime){

}
```

6. **Implement a function that reverses a queue recursively (you're not allowed other data structures, only have access to the queue STL functions - push(), pop(), front(), size()...)**

```
void reverseQueue(queue<int>& q){

}
```

7. **Implement a function that returns the number of even elements in a queue (you cannot use any additional data structures. Elements in the queue should as well be in the same order as when passed through the function after the function is done running)**

```
int countEven(queue<int> &q){

}
```

Sorting

1. **Merge, Quick, Shell, Bucket and Heap Sort (Implementations, Tracing and Time Complexities)**

2. **Sort the following array, [7, 4, 9, 13, 1, 5], using Merge, Quick, and Shell Sort**

3. **Sort the following array, [541, 847, 192, 316, 729], using Bucket Sort**

4. **Build a Min Heap from the following array, [6, 3, 5, 9, 8, 4]. After doing so, perform Heap Sort to it.**

Hashing

1. **Direct Hashing, Linear/Double Proving, Double Hashing and Separate Chaining (Implementations and Tracing)**

2. **Insert the following values into a Hash Table of size 5 using Double Hashing**

10, 20, 15, 30

Index = hash1() + (i * hash2()) % tableSize

hash1() → value % tableSize
hash2() → 3 - (value % 3)

0	**1**	**2**	**3**	**4**

Heaps

1. **Building Max/Min Heaps**

2. **Build a Min Heap from the following array, [10, 2, 5, 3, 7, 4]**

3. **Build a Max Heap from the following array, [2, 4, 7, 3, 8, 5]**