COSC 2436

Practice exam

<u>Question 1: Queues</u> - 18 Points

I. Jim is creating a music playlist which always plays songs in sequential order that was added to the playlist **first**. The playlist will have an indicator **rear** to mark the end of the list. Implement a function to REMOVE songs from the playlist after it is played. This function should have a constant time complexity **O(1)**

```
Struct Node{
  Node* next;
  string song;
  Node(string _song) : song(_song), next(nullptr) {}
  Node(string _song, _next) : song(_song), next(_next) {}
}
```

II. Write a function to implement a playlist enqueue system, where songs will be added to the playlist based on their priority. The playlist will also need a looping feature, after the last song is played, the playlist will return to the front.

```
Struct Node{
  Node* next;
  string song;
  int priority; //the lower the value, the higher the priority
  Node(string  _song, int _priority) : song(_song), priority(_priority), next(nullptr) {}
  Node(string _song, int _priority, _next) : song(_song), priority(_priority), next(_next) {}
}
//Write Enqueue Function Below
```

III. Write a program to simulate the Round Robin Scheduling using queues to hold the processes. For each process, the arrival time is given as well as burst time (execution time required by each process). And the fixed quantum time which is giving the processes a time slot to work on.

Using the given queue that holds all processes, write a function to simulate the Round Robin Scheduling that will output the order of Processes that are finished at the end.

Hint: Queue of all processes (given), queue of processing, queue of finished (output)

Process: P1, P2, P3, P4
Burst Time: P1(5), P2(3), P3(9), P4(5)
Arrival Time: P1(0), P2(0), P3(0), P4(0)
Quantum Time: 3

Output: P2, P1, P4, P3

Trace for partial credit.

Question 2: Heaps - 12 Points

I. Consider the following binary max heap: [25,14,16,13,10,8,12]. What is the content of the array after two delete operations? You may draw the tree or give the array representation if you'd like.

Question 3: Hashing - 19 Points

I. What would the hash table look like after inserting 19, 50, 89, 39 using linear probing? Use a table size of 5 and h(x) = x mod table size. Show all of your work for full credit.

II. Let us consider a simple hash function as “key mod 7” and sequence of keys as 50, 700, 76, 85, 92, 73, 101. What would the hash table look like after inserting all of these values using direct hashing? Use a table size of 7.

III. Suppose that we have a list of integers and would like to store them into a table. For each integer, we calculate some function h(x) = x mod table size, which will tell us at which position to store it in the table. If that position is already occupied (a collision), we will look over at the next position in the table and continue to do so until an empty location is found. What collision resolution method is this most similar to? Write a function that can insert a list of non-negative integers into a table using the described method. Assume an index is unoccupied if it's element is -1

You may start with the following if you'd like:

```
void insert(int A[], int arraySize){
Int hashTableSize = arraySize + 1;
int hashTable[hashTableSize];
//Finish Function Below
```

IV. Suppose that we have a list of integers and would like to store them into a table with the following constraints

a. Every piece of data (every integer) is valuable and cannot be lost.
b. We want to avoid having clusters in our table.
c. We want to limit/avoid using extra space.

Create a function that will insert a list of integers into a table while meeting these constraints.

Question 4: Sorting - 18 Points

I. Write a function for a recursive sorting algorithm with the following constraints

a. Sorting is done using a divide and conquer approach.
b. The time complexity is as follows:
   Best: O(nlog(n))
   Average: O(nlog(n))
   Worse: O(n^2)

II. Suppose that your program has to sort many items and you've implemented it using insertion sort. However, you find out that it's running very slow. This is probably due to the worst case of insertion sort. Write a new sorting algorithm function that is an enhanced/ optimized version of insertion sort since we need to reduce the number of swaps of the elements being sorted to minimize the complexity and time as compared to insertion sort to speed up our program.

III. Complete the sorting method below, this sorting uses divide and conquer approach to execute

```
void helper(int arr[], int left, int mid, int right)
{
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    i = 0; j = 0;k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Use the helper function to finish the Sorting function
void sort(int arr[], int left, int right) {
```

Question 5: Stacks - 20 Points

I. Write a function that evaluates a prefix expression with operators ‘+’ and ‘-’ (Assume you are given a string of single digits operators)
Correct output example: “+12” → 3
Correct output example: “” → 0
Correct output example: “-+432” → 5

```
int dequeue(string equation){
```

II. Suppose that you would like to write a program that checks if your expression parentheses are properly balanced/valid. For example, “[(1*3)+(5*4)]” is valid but “[(1+3)+[5*4)]” is not. Create a function that can take any string and return true if its parenthesis are valid or false otherwise.
The string will only contain parenthesis () [] {}, numbers, and operators + - * /.

```
bool isValid(string expression){
```

III. How can we determine whether a string is a palindrome with stack data structure? Implement a function that checks for palindromes using one or more stacks.

```
bool isPalindrome(string str){
```

IV. Write a program that can convert an expression like a+b*(c^d-e)^(f+g*h)-i to an expression like abcd^e-fgh*+^*+i-. To reduce the code, assume the only operators that we'll give you are '+' and ' -'.

```
Void convert(string str){
```

<u>Question 5: Recursion & Linked List</u> - 20 Points

```
void dequeue(int key){
// write your code here
```

I. Implement a recursive dequeue function that deletes a particular integer for the queue data structure. You'll have access to the queue, dequeue and front functions. Don't traverse the queue like you would a linked list

II. You're given a dynamic array of Linked Lists. That is, every element in the array is a linked list (contains a pointer to the head of the linked list). The array represents the day of the month, and each linked list will contain a list of foods that were eaten for that day. This is that person's meal plan for the day. For example,
"chicken salad" → "beef stew" → "tuna sub" → "tofu sub".
Write a function that takes in a food's name or its category as input ("salad" or "sub" for example), and removes anything containing that name from this person's meal plan

```
void removeFromMealPlan(LinkedList A[], int size, string name) {
// write your code here
```