Exam 2 Workshop Mini Assessment

SOLUTIONS

Stacks

---

1. 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 validParanthesis(string s){

```
stack<char> st;

for(int i=0; i<s.length(); i++){
  if( s[i]=='(' || s[i]=='{' || s[i]=='[' )
     st.push(s[i])

  else if( s[i]==')' ){
     if(st.empty() || st.top()!='(')
        return false;

     st.pop();
  }
```

→ repeat with "}" and "]" case

- note: on exam, actually write it out

```
  return st.empty()
}
```

Stacks

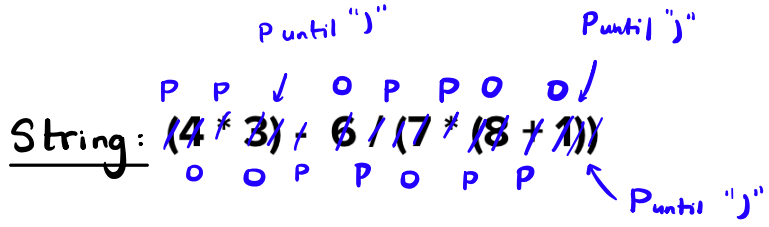
---

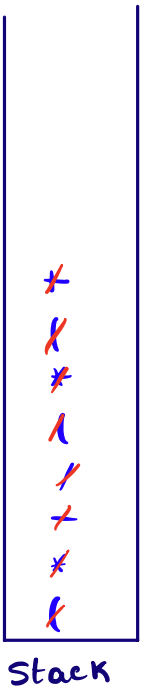
2. Convert the following infix expression to a postfix expression. Assume all integers are single digit integers.

P → push to stack

O → output

(4 * 3) - 6 / (7 * (8 + 1))





Output: 4 3 * 6 7 8 1 + * / −

Queues

---

3. Implement an enqueue() function which adds elements to the end of a “Queue” data structure (represented as a Linked List).

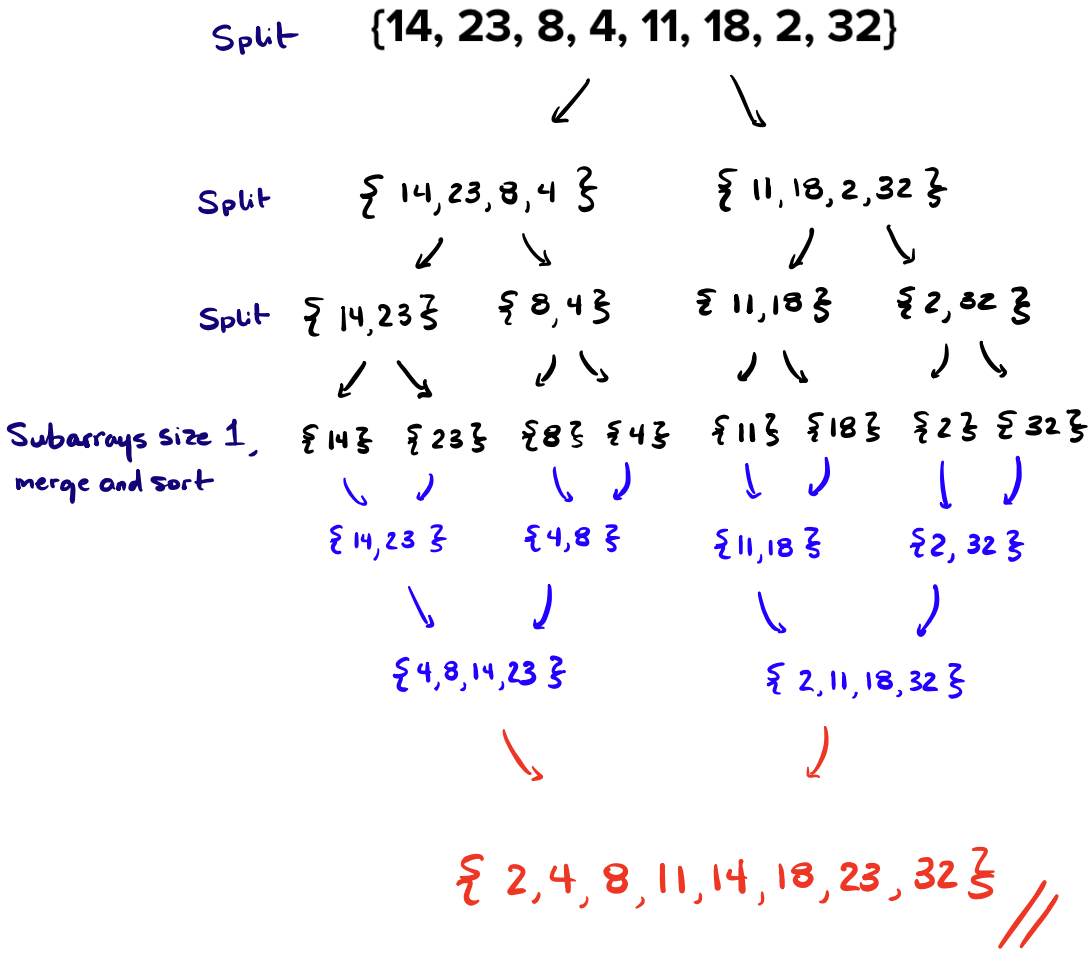
```
struct queueNode{
        int data;
        queueNode* next;

        queueNode(int value){
                data = value;
                next = nullptr;
        }
};

void Queue::enqueue(int value){

 queueNode *newNode = new queueNode(value)
 if( isEmpty())
    front = newNode;

  else {
     queueNode *temp = front;
     while( temp->next != nullptr )
         temp = temp->next;

      temp->next = newNode;
  }

}
```

Sorting

---

4. Sort the following array using Merge Sort

{14, 23, 8, 4, 11, 18, 2, 32}



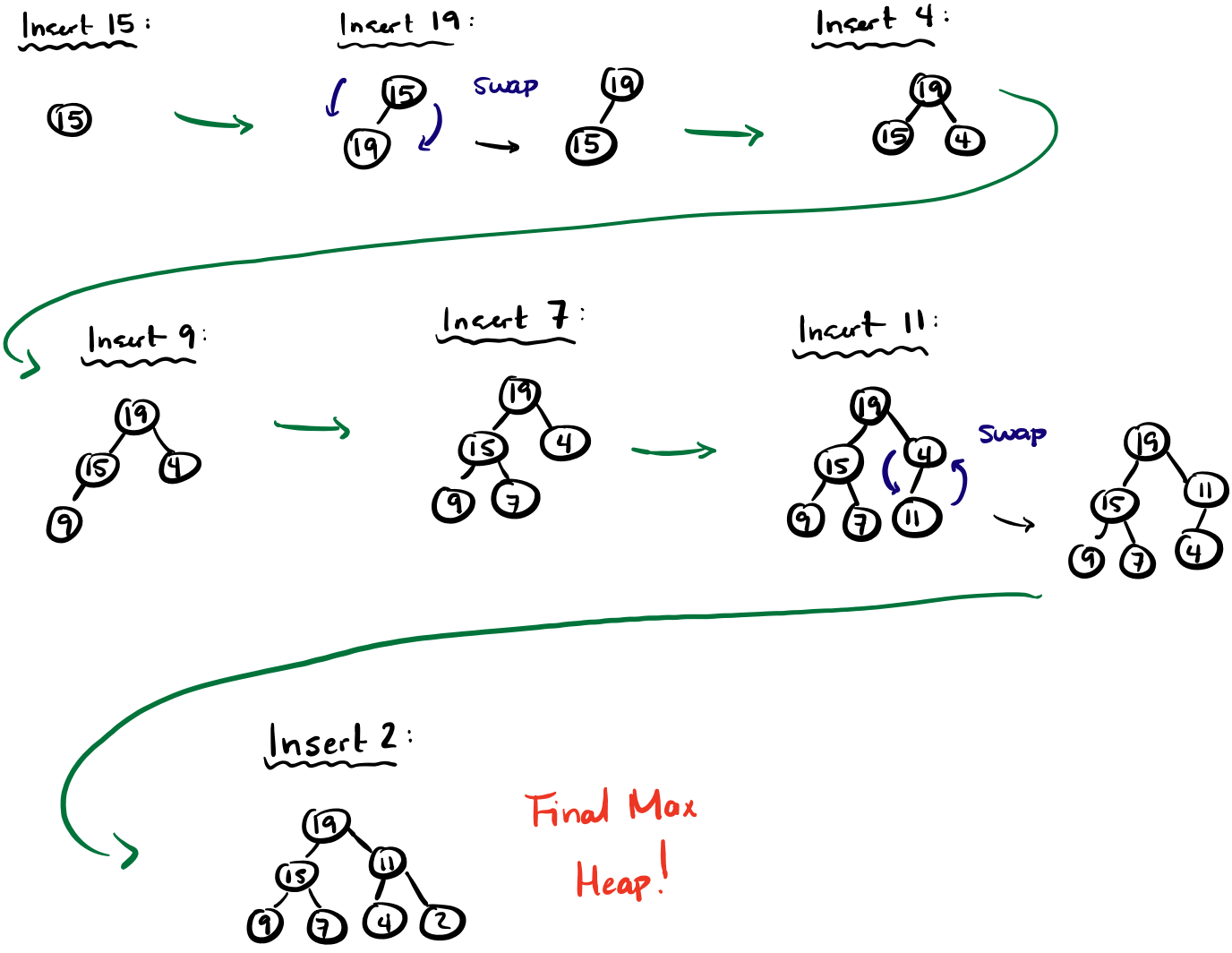
Heaps

---

5. Build a max heap from the given array, **{15, 19, 4, 9, 7, 11, 2}.**

Show what the heap looks like (as a tree) after every insertion.

~~{15, 19, 4, 9, 7, 11, 2}.~~



Hashing

---

6. Insert the following values, **{13, 26, 8, 3, 31, 14}** into a hash table of size 10 using Linear Probing.

```
int hash(int x) {
        return x % 10;
}
```

0	1	2	3	4	5	6	7	8	9
∅	31	∅	13	3	14	26	∅	8	∅

~~{13, 26, 8, 3, 31,~~ 14}

13 → 13 % 10 = 3

26 → 26 % 10 = 6

8 → 8 % 10 = 8

3 → 3 % 10 = 3 (not available, find next available index)
↳ 4

31 → 31 % 10 = 1

14 → 14 % 10 = 4 (not available, find next available index)
↳ 5
