CSC 212: Mock Exam 1

October 13, 2022

Name: ____

Instructions

- Write your name at the top of each page in case the pages become separated.
- Answer the questions in the space provided. If you run out of room, continue on the back of the page.
- Questions asking you to **explain** or **define** something should be answered in complete sentences. (1-2 sentences will likely suffice for a complete explanation.)
- Questions asking for code should be written in something approximating C++. Your code does not need to be perfect, but it should be clearly recognizable as C++.
- You may use **one 8 1/2 x 11in** page of notes. If you do, you must hand in your note page with th exam.

Question	Points	Score
1	0	
2	0	
3	6	
4	0	
5	0	
6	0	
7	0	
Total:	6	

Throughout the exam, assume that we have the following `Node` struct defined:

```
struct Node {
    int n;
    Node* next;
}
```

1. Explain briefly what a memory leak is.

2. Explain the difference between $O(f(n))$ and $\Omega(f(n))$. (If you wish, you may give precise definitions of both rather than writing a sentence.)

3. Suppose we have the following `Queue` class defined:

```
class Queue {
    public:
    Queue();
    ~Queue();
    void Enqueue(int n);
    int Dequeue();
    int Size()
};
```

Using **only two** `Queues` complete the implementation of the following `Stack` class. You do not need to worry about the case where someone tries to pop from an empty stack.

```
class Stack {
  private:
  Queue *q1;
  Queue *q2;

  public:
    Stack() {
      q1 = new Queue();
      q2 = new Queue();
    }

    void Push(int n);

    int Pop();
};
```

(a) (3 points) `void Stack::Push(int n) {`

```
}
```

(b) (3 points) `int Stack::Pop() {`

```
}
```

4. (a) Implement a function that returns `true` if every element in a linked list in even and `false` otherwise. If the list is empty, the function should also return `true`. (Remember that `Node` was defined at the start of the exam.)

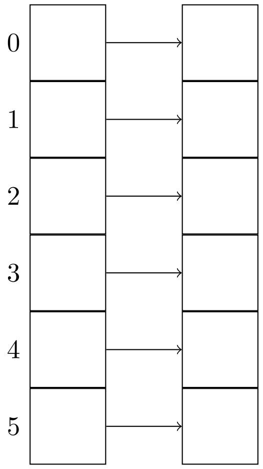
```
bool AllEven(Node *head) {

}
```

(b) What is the time complexity (big-O) of your solution (in terms of the number of elements n in the list)? Why?

5. Suppose we have a hash table with $m = 6$ and $h(k) = k \bmod 6$. We will insert the following elements into the table:

$$6, 3, 0, 4, 7, 1$$

(a) Suppose we have an implementation that uses chaining to resolve collisions. Draw the state of the chains after inserting the elements above. (**Note:** There are six elements to insert. You will not use every box in the picture below. Simply leave any extras blank.)



(b) Suppose we have an implementation that resolves collisions using *linear probing*. Draw the state of the backing array after inserting the elements above:

0	1	2	3	4	5

6. A string is said to be a *palindrome* if it is the same forwards and backwards. (For example, *racecar* is a palindrome. *apple* is not. Implement the function `IsPalindrome` which returns `true` if the string is a palindrome and `false` if not. You may assume the existence of the following `Stack` class, but you are not required to use it. (There are a number of possible solutions to this problem.)

```
class Stack {
  public:
    // pop the top element from the Stack
    char Pop();
    // "peek" at the top element of the stack, but *do not* pop it
    char Peek();
    // push an element onto the Stack
    void Push(char c);
    // return the number of elements on the Stack
    int Size();
}
```

```
bool IsPalindrome(std::string s) {

}
```

7. Suppose we have a sorted array of distinct, non-negative integers. Find the smallest non-negative integer that is *not* in the array. For example:

   - The smallest missing integer in $[0, 1, 2, 3, 6, 7, 9, 10]$ is 4.
   - The smallest missing integer in $[1, 2, 4, 5]$ is 0.

   Write a function that returns the smallest missing integer from such an array and returns -1 if there is no missing element. Your function should have time compexity $O(\log n)$.

```
int SmallestMissing(std::vector<int> arr) {

}
```