This problem set has 10 questions, for a total of 100 points. Please carefully read the guidelines below:

- Provide your final answers only within the designated spaces on each of the questions. We use automatic grading for some questions – answers outside these spaces will not be graded.
- You may annotate your answers digitally on the PDF, or alternatively, you can print the PDF and write your answers by hand. If you choose the second option, please ensure that your handwriting is legible, and the software/hardware used for scanning the document does not change the original format of the PDF, keeping the same structure, orientation, and page size. Once you are done, submit your file through Gradescope.
- We expect you to focus on fully understanding how to solve each problem, not just on obtaining the final answer. This is important not only for your learning, but also for your performance in the course, as similar questions may appear in your exams. Do not hesitate to ask for help if you have any questions.

Your Name:

Queues

1. (10 points) Write a function (use C++) that takes a queue of integers and returns the *largest* element in the queue. The function should match the following signature: `int find_max(std::queue<int>& Q)`. You should not use any other data structure (e.g., stack, array, etc.) to store the elements of the queue. After returning from the function, the elements of the queue should remain in the same order as before the function call.

2. (10 points) Assume the sequence of elements $1, 3, 6, 7, 10$ has been pushed into a queue `Q` in that order. Determine the output of the `mystery` function below, when called with `Q` as the argument.

```
int mystery(std::queue<int>& Q) {
    int result = 0;
    int loop = Q.size();
    for(int i = 0 ; i < loop ; i++) {
        if(!(i % 2)) {
            result += Q.front();
        }
        else {
            result *= Q.front();
        }
        Q.pop();
    }
    return result;
}
```

2. _____

Priority Queues

3. (10 points) Consider the insertion of the following values (in the given order) into an initially empty *max-heap*: $31, 5, 8, 23, 17$. Write the status of the heap array after each insertion (i.e. draw a sequence of arrays).

4. (10 points) Take the following sequence and apply the `buildHeap` algorithm: [‘stack’, ‘queue’, ‘deque’, ‘list’, ‘vector’, ‘array’, ‘string’, ‘map’, ‘set’]. Draw the tree representation of a Min-Heap after the last iteration of the algorithm.

Linked Lists

5. (10 points) Write a function (use C++) that counts the number of triplets (three adjacent elements) whose sum is equal to 0. The function should match the following signature: `int count_triplets(Node *head)`. where `head` is a pointer to the first node of a doubly linked list. Assume the list has at least 3 elements.

6. Assume each pointer uses 8 bytes, each integer uses 4 bytes, all linked lists are implemented with a `head` and a `tail` pointer, and each node in a linked list stores an integer.

   (a) (5 points) How many bytes are necessary to store a Singly-Linked List of length 100?

   (a) _____

   (b) (5 points) How many bytes are necessary to store a Doubly-Linked List (DLL) of length 100?

   (b) _____

Recursion

7. (10 points) Which of the following descriptions best describes what `mystery` does?

```
int mystery(int *arr, int n) {
    if(n == 1) {
        return arr[0];
    }
    int val = mystery(arr + 1, n - 1)
    return (arr[0] < val) ? arr[0] : val;
}
```

   A. find the minimum element in `arr`

   B. find the maximum element in `arr`

   C. find the the sum of all elements of `arr`

   D. sort all elements of `arr`

   7. _____

8. (10 points) Write a recursive function (use C++) that returns the sum of all even numbers of an input array. The function should match the following signature: `int sum_array(std::vector<int>& arr, int n)`, where `n` indicates the number of elements in the array.

9. (10 points) Write a recursive function (use C++) that reverses the elements of an input array **in place**. The function should match the following signature: `void reverse_array(std::vector<int>& arr, int n)`, where `n` indicates the number of elements in the array. The function should modify the array in place and perform no more than $n/2$ swaps.

10. (10 points) Given the following recursive function. Draw the *recursion call tree* (graphic representation of all the calls), for an initial call of this algorithm with `m=14` and `n=4`. The function is a recursive implementation of Dijkstra's algorithm for finding the greatest common divisor (GCD) of two numbers.

```
int gcd(int m, int n) {
    if (m == n) return m;
    else if (m > n) return gcd(n, m - n);
    else return gcd(m, n - m);
}
```