EE599 - HW6
Computing and Software for Systems Engineers

- Unless specified: for each question that you write code:
    - Provide GTest.
    - Provide runtime analysis.
    - Proof of correctness is not necessary unless specified.
- For submission, please create a zip file of all of your assignments and only submit one file.
    - PLEASE REMOVE ALL FOLDERS STARTING WITH **bazel-*** before submitting. **To do this run : `bazel clean`**
    - Leave any extra instructions for the graders in a README text file.
    - Our grader should be able to call blaze run/test ... and run your code/test.
    - **For the purpose of faster grading, please put your code in one solution.cc / solution.h, test.cc, and main.cc.**
- Deadline: Wednesday, March 4, before 6pm.
- Total: 120 points. 100 points is considered full credit.
- **Academic Conduct**
    - Students are encouraged to collaborate on general solution strategies for homework. The writeup, however, must be your own - **you may not copy someone else's solution**.
    - In addition, your homework should list all the fellow students with whom you discussed the solutions.

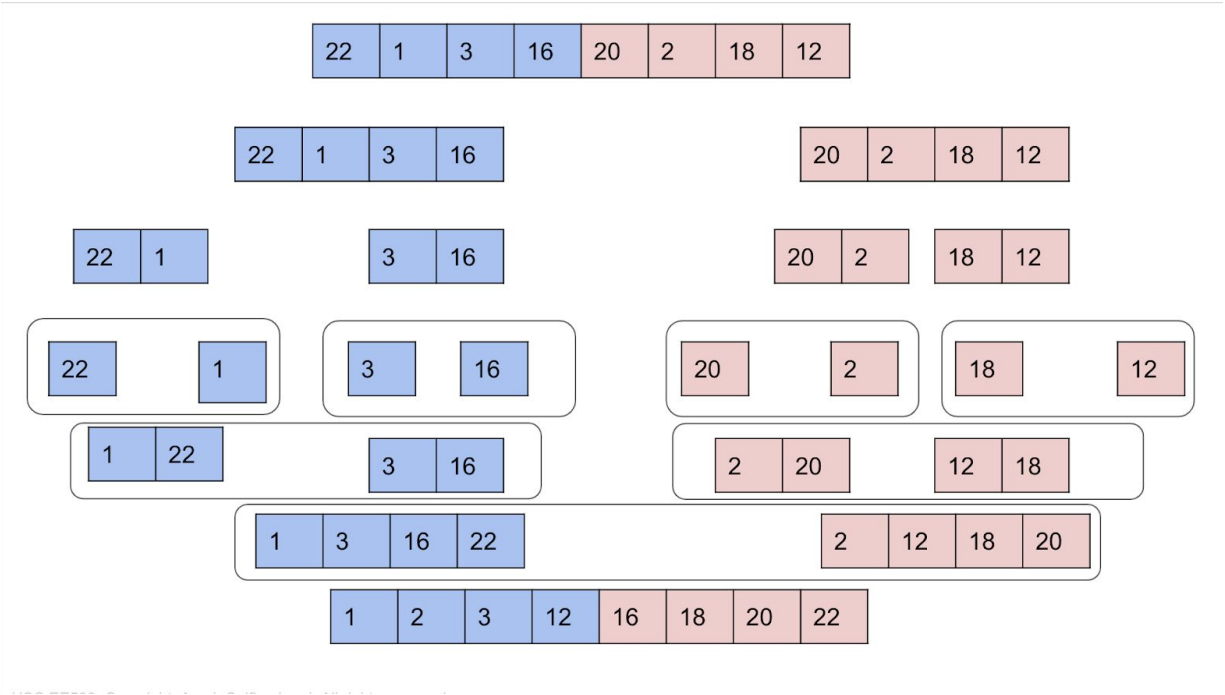
Question 1 (20 Points. Easy)

Complete or select the correct word in the following statements:

- A tree is an **undirected/directed** graph.
- A tree is a **connected / unconnected** graph.
- A tree is a **cyclic / acyclic** graph.
- In a tree, there **is / is not** a path from each vertex to all other vertices.
- A simple graph is a graph that _____.

Question 2 (20 Points. Easy)

This picture shows the order of dividing and merging in MergeSort:



Please create a similar image for sorting the letters in the following string:

“ilovecoding”

- Please clearly specify when you are dividing and when you are merging and what two arrays are being merged.
- You can either attach an image OR text representation of your solution.

Question 3 (20 Points. Medium)

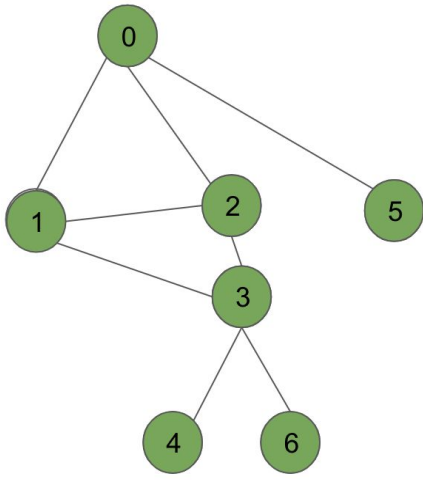
Consider the following representation of a graph using an std::map that maps each vertex to its neighbors, and the sample usage of it:

```
class Graph {
public:
 Graph(std::map<int, std::set<int>> &vertices) :
v_(vertices) {}
 std::map<int, std::set<int>> v_;
};

int main() {
 std::map<int, std::set<int>> vertices{
     {1, {2, 3}},
     {2, {1, 3, 4, 5}},
     {3, {1, 2, 4}},
     {4, {2, 3, 4}}
 };
 Graph g(vertices);
}
```

- Add a new method to this class to implement the **non-recursive version** of **DFS** algorithm.
- Your function should take a vertex called root as the starting point, and should output a vector containing the nodes that it visits in DFS order.

Example:



Input: The above graph and 0 (root)

output: [0, 1, 2, 3, 4, 6, 5]

Note that an undirected graph shown above can be converted to a directed graph by adding two opposite edges between each two nodes that are neighbors in the directed graph.

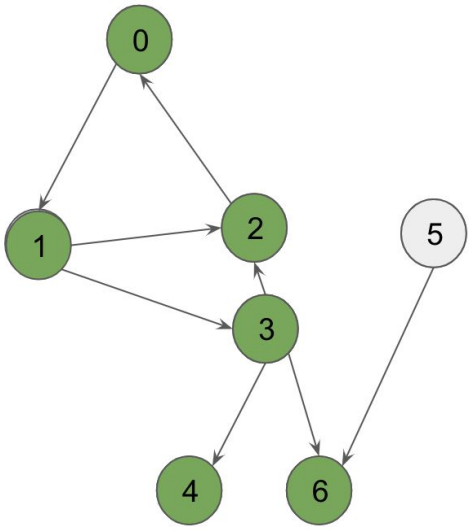
Question 4 (20 Points. Easy)

For the Graph class in Question 2, please write a method called DFS_ALL(), where it ensures that **all** nodes are visited and returns the visited nodes in DFS order in a vector.

- Note: DFS_ALL() doesn't have any input parameters.

Example:

Input:



output: [0, 1, 3, 2, 4, 6, 5]

Question 5 (20 Points. Medium)

You are given a maze, a start and an end point, find out if there is at least one path from start to end.

- The maze is a grid where some cells are blocked and you cannot go there.
- You can only move to the right, left, down, or up. You cannot move diagonally..
- The maze is represented by a 2x2 vector where a 0 represents a blocked and 1 represents a free cell.
- Start and End are specified as pairs: (i,j), where i is the row, and j is the column.
- Notice that you don't need to generate the path. Just find out if at least one path exists.

Example 1:

Start				
				End

The maze is represented as a 2x2 vector called **maze** as follows:

1	1	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	0	1
1	1	1	1	**1**

Input: 2x2 maze (as shown above), start=(0,0), end=(4,4)

Output: true (because there is a path from cell (0,0) to (4,4)

Example 1:

		Start		
				End

Is represented as a 2x2 vector **maze** as follows:

1	1	0	0	0
1	1	**1**	1	1
0	1	0	0	1
1	0	0	0	0
1	1	1	1	**1**

Input: 2x2 maze (as shown above), start=(1,2), end=(4,4)
Output: false (because there is no path from cell (1,2) to (4,4)

Question 6 (20 Points. Medium)

Given a vector of integers **v**, and an index i in the vector, rearrange the vector such that all items less than or equal to v[i] are on the left, and all items greater than v[i] are on the right side.
Note:

- The left and right side don't need to be sorted.
- You can assume **i** is always within v's boundaries.
- Your algorithm should be O(n)

Example:
Input: v= [9, 7, 5, 11, 12, 2, 14, 3, 10, **6**], i=9.
Output: [5, 2, 3, **6**, 12, 7, 14, 9, 10, 11]

Note that there are multiple correct outputs. For example, the following is also a valid output:
Output: [5, 3, 2, **6**, 14, 7, 12, 9, 10, 11]

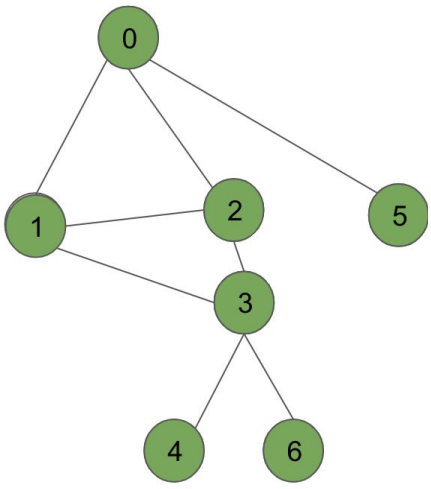
Optional Questions

The goal of this section is to introduce you to more challenging questions and some common problems in coding and algorithms.

- These questions don't have any credits.
- We may not provide complete solutions or grading for them.
- Solving them is completely optional.

Question 1

Given an undirected graph, how to check if there is at least one cycle in the graph?
For example, the following graph has a cycle: 0, 1, 2:



- Your function should only return true/false indicating if the graph has a cycle. You don't need to actually print the cycle.