EE599 - HW4
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.
- Deadline: Tuesday, Feb 18th, before 6pm.
- Total: 150 points. 120 points is considered full credit.

Question 1 (10 Points. Easy)

- **Filter:** Write a function that filters out the odd numbers and keeps the even numbers in a vector (use std::copy_if) and returns a new vector that contains the results.
- **Map**: Write a function that takes a vector of integers as input **in** and outputs a new vector **out** where out[i] = in[i] ^ 2 (Use std::transform)
- **Reduce**: Write a function that sums up all elements in a vector (Use std::accumulate)

Note: for copy_if you need to use std::resize to resize and shrink the output vector. See the example here.

Question 2 (40 Points. Medium)

Please implement the following class for MaxHeap:

- Provide GTest only for methods that are marked with "GT".

```cpp
// Only methods which are marked by "GT" should be tested
class MaxHeap {
public:
 MaxHeap();  // default constructor

 int GetParentIndex(int i);     // *GT*
 int GetLeftIndex(int i);       // *GT*
 int GetRightIndex(int i);      // *GT*
 int GetSmallestChildIndex(int i);  // *GT*


 int GetLeft(int i);
 int GetRight(int i);
 int GetParent(int i);


 int top();  // GT
 void push(int v); // GT
 void pop(); // GT
 void TrickleUp(int i);
 void TrickleDown(int i);

private:
 std::vector<int> data_;
};
```

Question 3 (40 Points. Medium)

Please implement the following class for a Binary Search Tree (BST):

- Only methods that are marked with "GT" should be tested.

```cpp
struct TreeNode {
 int val;
 TreeNode *left;
 TreeNode *right;
 TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};


class BST {
public:
 BST();

 // Inserts elements of initial_values
 // one by one into the Tree
 BST(std::vector<int> initial_values);
 ~BST();

 void push(int key);   // **GT**  Inserts a key inside Tree
 bool find(int key);   // **GT** Returns true of key is in the tree
 bool erase(int key);  // **GT** Removes the key from the tree. If
not successful, returns false.


private:
 TreeNode *root_;
}
```
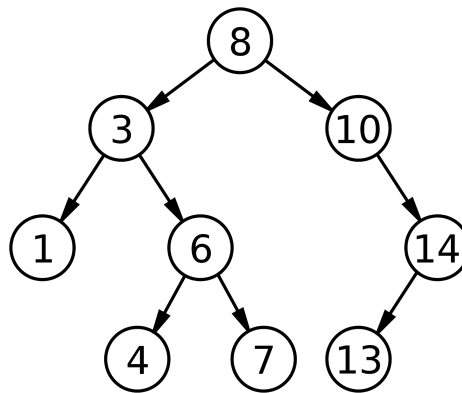
Question 4 (20 Points. Medium)

For the BST in the previous question, write a function that will traverse the tree level by level. That is, the root is visited first, then the immediate children of the root, then the grandchildren of the root, and so on.

- **Example: Input:**

Output: 8 3 10 1 6 14 4 7 13

👉**Hint:** Use std::queue to keep track of the children of a node until it is time to visit them.

Question 5 (20 Points, Easy)

Write a function that implements heap-sort on a vector of integers. The function's return type should be void and the same input vector should get modified and sorted.
You should use std::priority_queue to implement your function.

Sample input: [5, 9, 3, 1, 7], output: [1, 3, 5, 7, 9]

Question 6 (20 Points, Easy)

Find the k(th) largest element in an unsorted vector and return that value.
- You should do this without sorting the vector.
- You can assume the input vector does not have duplicate values.

Example 1:

Input: [0,2,1,5,6,3] and k = 2

Output: **5** (because the largest is **6**, and the second largest is **5**)

Example 2:

Input: [-2, 3,-1,2,5,6,10] and k =3

Output: 5 (because the largest is 10, 2nd largest is 6, and 3rd largest is 5)

👉**Hint:** Heap heap hurray! (Use STL containers)

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

Implement the non-recursive version of in-order traversal of a tree, i.e. your function should not call itself directly or indirectly.

👉**Hint:** Use std::stack.

Question 2

Given a binary tree in a vector, determine if it is a valid binary search tree (BST).

Assume a BST is defined as follows:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.

- Both the left and right subtrees must also be binary search trees.

Question 3

Given the in-order traversal and the post-order traversal of a binary tree, recover the original tree.

Sample input: in-order [1, 3, 4, 6, 7, 8, 10, 13, 14], post-order [1, 4, 7, 6, 3, 13, 14, 10, 8]
Output: