Problem Number(s)	Course Evaluation	Possible Points	Earned Points
1	Sorting	10	
2	Binary Tree	10	
3	BST	20	
4	AVL	20	
5	Binary Heap	20	
6	Priority Queue	20	
7	Binomial Queue	20	
		TOTAL POINTS 140	

Instructions:

i. **This is a sample exam, so number of questions here and the points per each question does not reflect the actual exam.**

ii. **You have 1 hour to complete the Exam.**

iii. If information appears to be missing from a problem, make a reasonable assumption, **state it and proceed.**

iv. If the space to answer a question is not sufficient, use the back of each question's page.

v. **The exam is closed book. No calculators allowed.**

1. SORTING

Consider an array, **SORT,** shown below

SORT =

E	A	S	Y	Q	U	E	S	T	I	O	N

Assuming that the elements are to be arranged in an ascending order starting from the leftmost element, use this array to provide the solution for the following questions.

a) Show the first partition achieved using the Quick sort algorithm. Assume the left-most element (i.e. index 0) is selected as the pivot [2 points].

b) Show the step by step process of sorting the above array using the Heapsort algorithm: Please show the resulting array for the first 3 steps after the min heap is created. [5 points].

c) In class we looked at sorting using shell sort algorithm. Wikipedia describes Shell sort as "a generalization of insertion sort that allows the exchange of items that are far apart. The idea is to arrange the list of elements so that, starting anywhere, considering every h^{th} element gives a sorted list. Such a list is said to be *h*-sorted". Starting with SORT as show, show the sorting achieved with decreasing h (gap length as the sorting continues). Please label the gap length used in each sorting iteration [3 points]:

SORT	E	A	S	Y	Q	U	E	S	T	I	O	N
h – 5												
h - 3												
h - 2												

2. Trees

A tree definition is given as follows:

```
typedef struct binaryNode*   BinaryNode_Ptr;
struct binaryNode
{
   BinaryNode_Ptr left;
   BinaryNode_Ptr right;
   int item;
};
```

A pointer to the root of the tree is declared as follows:

```
BinaryNode_Ptr root;
```

Write a recursive function, **destroy**, that frees all memory held by the nodes of the tree. [10 points]

https://www.geeksforgeeks.org/delete-linked-list-using-recursion/

```
/* Recursive Function to delete the entire linked list */
void deleteList(struct Node* head)
{
    if (head == NULL)
        return;

    deleteList(head->next);

    free(head);
}
```

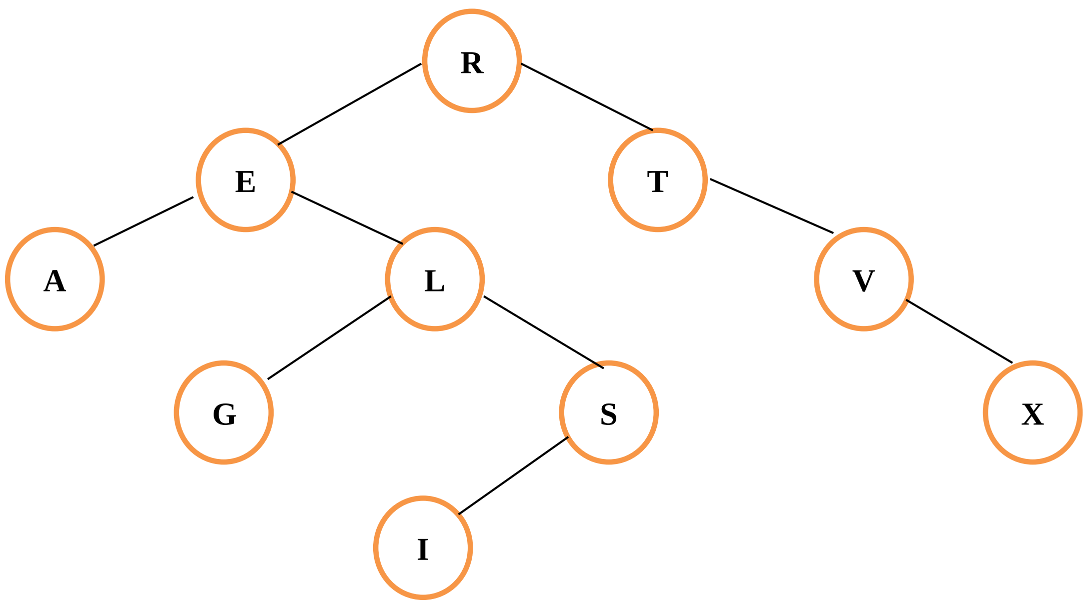
```
// Iterative funtion

void destroy_list(Node** pHead)
{
    Node* temp;
    while (*pHead != NULL)
    {
        temp = *pHead;
        *pHead = (*pHead)->next;
        free(temp);
    }
}
```

3. Binary Search Tree

a) Briefly explain what a binary search tree (BST) is, listing its properties [3 points]

b) Is the following binary tree a BST or not, and why (assume ASCII values for the alphabet letters (e.g. A=26))? [2 points]



c) If you answered (No) on (b) above, re-draw the tree in correct ordering [2 points]

d) Describe an optimally efficient algorithm to find the predecessor of a given node *n* in a BST and explain why it works [3 points].

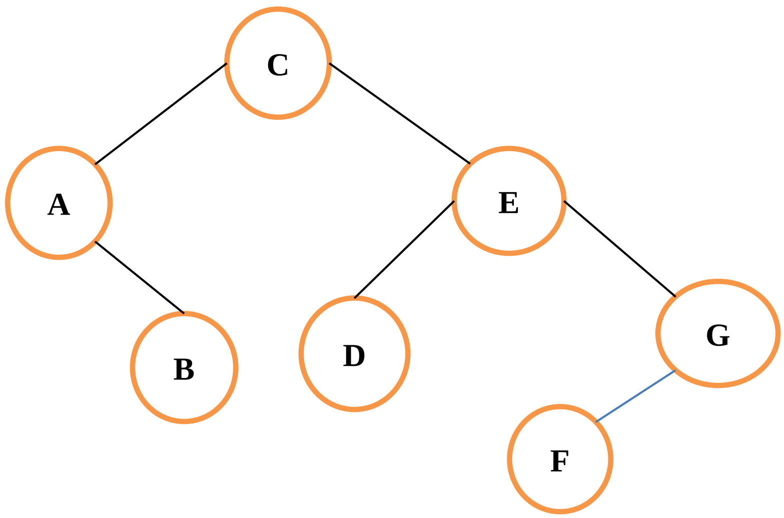
e) Describe an optimally efficient algorithm for deleting a node *d* from a BST when neither of *d's* subtrees is empty. Explain why it works and prove that what remains is still a BST [5 points].

f) Assume that node *l*, whose key is k_l , is a leaf of a BST and that its parent is node *p*, with key k_p. Prove that, of all the keys in the BST, k_p is either the smallest key greater than k_l or the largest key smaller than k_l. [5 points]

4. AVL Tree

a) For each node shown in the binary tree below, show its depth, height and the AVL balance factor. Write your answers in the given table [5 points]

NB: An incorrect answer will attract -1 point



Node	**Depth**	**Height**	**Balance Factor**
A			
B			
C			
D			
E			
F			
G			

a) What is the main advantage of an AVL Tree over a Binary Search Tree (BST)? [5 points]

b) Draw the sequence of AVL trees obtained when the following keys are inserted one-by-one, in the order given into an initially empty AVL search tree: {F, E, A, B, D, C, G}. Identified rotations, if there is any [10 points].

5. **Binary Heap**

a) Explain what the heap data structure is, state its defining properties and explain how to convert between the tree and vector (array) representations of a heap. [2 marks]

b) Describe an optimally efficient algorithm for transforming any random vector into a binary heap vector and explain why it works. [4 marks]

c) Using the tree instead of the vector representation for clarity, apply this algorithm to the binary tree isomorphic to the letter vector “P I S K T Z O P V N”, producing a frame-by-frame trace of the execution. For this answer, please show a new tree whenever any nodes change. [5 marks]

d) Explain how to rearrange the heap after having extracted its top so that what remains is still a heap. Follow this procedure to extract the top three values, one by one, from the heap you built, producing a frame-by-frame trace as above. [5 marks]

e) Describe a way to insert a new value into an existing heap in time O(log n) where n is the heap size.[4 points]

6. **Priority Queue**

a) What is a priority queue? Explain the data structure known as a binary heap and document how a heap is stored in a simple linear block of memory. [4 points]

b) If a binary heap stores N items, describe how it can be viewed as an almost-balanced binary tree. What difference can there be between the greatest and least lengths of paths from the root of the tree to a leaf? What operations must be performed to move from one node in the tree to (a) its parent and (b) its offspring? [5 points]

c) Describe, and estimate the costs of, procedures to:

   i. Insert a new item into an existing heap [2 points];

   ii. Delete the topmost item from a non-empty heap [2 points];

   iii. Starting from an array holding N items in arbitrary order, rearrange those items so that they form a heap, taking time less than that which would be needed if the items were just inserted into the heap one after the other[2 points];

d) A stable sorting method is one where items whose keys compare as equal will appear in the output in the same order that they appeared in the input list. Would a heap sort based on the algorithms you have documented be stable? Justify your answer [5 points].

7. **Binomial Queue**

   A binomial queue is implemented as a Max-Heap.

   a) Give the binomial queue that results when the keys E A S Y are inserted into an initially empty binomial queue (Show a frame by frame sequence) [4 points].

   b) Give the binomial queue that results when the keys Q U E S T I O N are inserted into an empty binomial queue [6 points].

c) Give the result of delete the maximum for each queue [4 points]

d) Give the results when the "merge/join" operation is performed after part c above is completed. [6 points]