Name: _____ NetID:_____

Instructions

I wish you good skill! Read these instructions FIRST!

Please put your name/NetID on this page and at least your NetID on the rest of the pages.

Please be smart in your time management. This is an assessment.. I want to understand your level of understanding. Don't get stuck on a question. Do the questions you are most sure about first.

Read the question. Then read it again! Try to be sure what I'm asking you to do. When you get done, read the question yet again and see if you answered it completely.

Be clear, be complete. Be concise. Answer the question, but don't try to tell me everything about everything.

If you make assumptions, write them out.

Show your work, it helps me give you partial credit even if you get lost.

You will have the full three hours to complete the exam. When you are done, come drop your exam paper in the box up front.

Desks clear. No talking. No notes or electronic devices will be allowed.

For all code, you may assume that we are `using namespace std;` and that we've done

- `#include <iostream>`
- `#include <string>`
- `#include <vector>`
- `#include <cmath>`
- `#include <stdexcept>`

If the exam is not on the front table in the box when I call time, then you will receive a zero. Please understand that I truly mean this – I will not tolerate anyone trying to extend the exam. It is not fair to those students who finish the exam on time.

Please initial here to indicate that you understand this: _____

Name: _____ NetID:_____

1. Short answer questions
   a. If I have 127 keys in random order, what is the height of the shortest binary search tree (BST) that could possibly contain all keys?

   b. If I have 127 keys in random order, what is the height of the tallest binary search tree that might result?

   c. If I have n keys in random order, what is the expected height of the BST?

   d. I have a thousand keys in unknown order stored in a linked list. I want to find the node containing the key `foobar`.
      i. What is the fewest number of comparisons required to find it? Why?

      ii. What is the maximum number of comparisons required? Why?

   e. I have a thousand keys stored in a sorted array. I want to find the node containing the key `foobar`.
      i. What is the fewest number of comparisons required to find it? Why?

      ii. What is the maximum number of comparisons required? Why?

   f. If I have an $\boldsymbol{O}(n^2)$ algorithm and running a job with 10 items takes 10 seconds; about how long would it take to run a job with 20 items?

   g. On Linux systems, if you lose your password, the sysadm cannot recover it for you and you need to generate a new one. You know that it has something to do with hashes. How does password checking work here?

   h. I represent a unweighted graph with GRAPHSIZE nodes using an adjacency matrix. Write the C++ data structure to represent the matrix (just the data, not the functions). How big is it?

```
const unsigned GRAPHSIZE = 200;
```

Name: _____ NetID:_____

2. You need to write a function that reads in some data from a file and fills in a std::vector<int>. Pass the filename and vector into the function as parameters. Write me the **header** for this function (i.e. what you would put into the .h file).

3. Your code is ready to read a couple of integers `x` and `y` from the `input` filestream which you've opened to a file. You're not sure if they are there or if they are formatted correctly. What code do you propose to read them in?

```
int x = 0;
int y = 0;
std::ifstream("foo.txt");
```

Name: _____ NetID:_____

4. Short answer big ***O*** notation

a. `linearSearch(A, key)`: Time to search an array of size n for a key

b. `binarySearch(A, key)`: Time to search a sorted array of size n for a key

c. `bubbleSort(A, key)`: Time to sort an array of size n in place

d. `quickSort(A, key)`: Time to sort an array of size n in place

e. `radixSort(A, key)`: Time to sort an array of size n zip codes in place

f. `BST::insert(key)`: Time to insert a key into a binary search tree of size n

g. `AVL::insert(key)`: Time to insert a key into a AVL tree of size n

h. `HASHTABLE::contains(key)`: Time check for a key in a hash table of size n

5. I have a linked list of n keys from which I'll do a traversal. If the key is in a hash table, I'll add it to a binary search tree.

```
for(const Node* p = head; p != nullptr; p = p->next) {
    if ( hash_table.contains(p->key) ) {
          search_tree.insert(p->key);
    }
}
```

What is the big ***O*** in the above?

Name: _____ NetID:_____

6. My company has some important codes that it is developing. I've done the analysis on some of them. On others, I've just done some test runs on the prototypes. Fill in all the blanks in the tables below with your best guess as to their value.

P1	O(n)
size	Time in secs
10	17
20	
100	
1000	

P2	O( )
size	Time in secs
6	0.5
12	2
60	50
	200

P3	O( )
size	Time in secs
4	1
16	2
64	3
256	

7. My company needs P2 to run overnight (from midnight to 6am). In P2's case, n represents the number of customers. I want to know **about** how many customers I can support with this program (i.e. what is the largest that n can be and finish in 6 hours = 21,600 seconds).

Name: _____ NetID:_____

8. Write a C++ class that implements a queue. You may use inline code. Use good judgment for coding practices. It is not necessary to implement a copy constructor or an assignment operator. Use the continuation page if necessary. It should work for ints, doubles, and strings. It supports the following methods:

- constructor(unsigned maxSize) – the do not exceed/max size allowed for the queue.
- enqueue(x) – add an item at the end of the queue
- dequeue() - removes the lead item
- peek() - returns a reference to the lead item (but does not remove it)
- isEmpty() - true if the queue is empty, else false

Name: _____ NetID:_____

(more space for question 8 if needed)

Name: _____ NetID:_____

9. I have this class for a binary search tree of integers. Please write just the insert function.

```
class IntBST {
private:
  struct Node {
    int key;
    Node* left;
    Node* right;
    Node(int k) : key(k), left(nullptr), right(nullptr) {}
  };
  Node* root_;
public:
  IntTree() : root_(nullptr) {}
  // destructor, search, etc.. not shown
  void insert(int key); // Please write this one
};
```

Name: _____ NetID:_____

10. Some heap questions. Answer concisely, but completely.
    a. What is the definition of a heap?

    b. What properties does it have?

    c. How is it best represented?

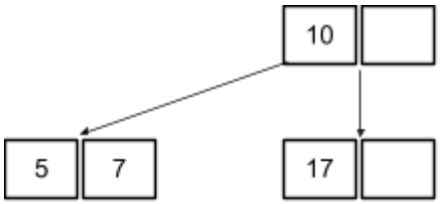
    d. What are some good uses for a heap?

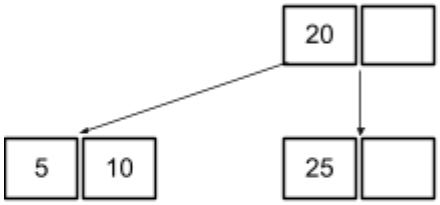
    e. How can I use a heap to sort things (i.e. describe heap sort)

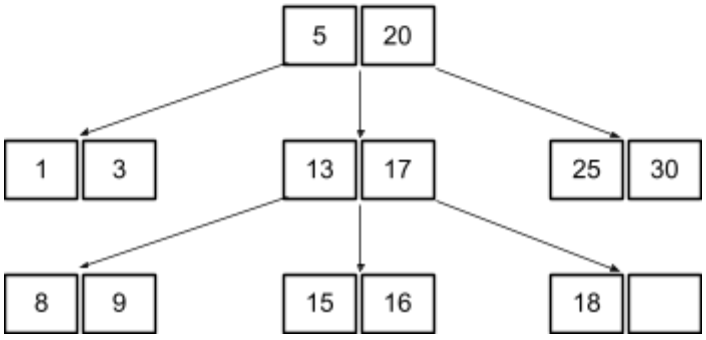
    f. What is the big ***O***() cost for a heapsort?

Name: _____ NetID:_____

11. For each of the 2-3 trees below on the left, Insert the value 12 and neatly draw the resulting tree to the right.

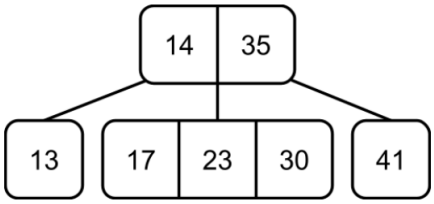




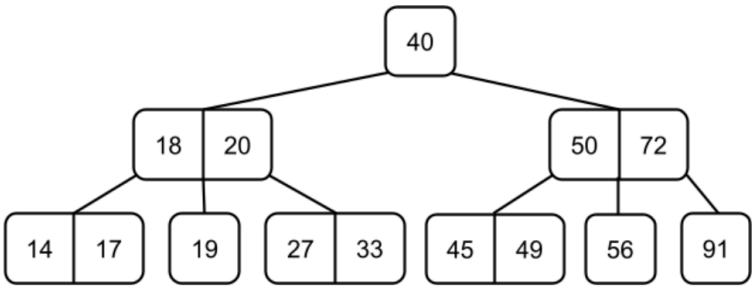


Name: _____ NetID:_____

12. Time to work on some 2-3-4 trees… First, draw the tree after a right rotation at the (17, 23, 30) node.



And do right rotation at (18,20) below



Name: _____ NetID:_____

13. AVL trees are another kind of auto-balancing tree that we studied. They are like super binary search trees!

a. What special properties do AVL trees have that make them better than generic binary search trees.

b. One important measurement is the “balance factor.”
   i. How is it computed?

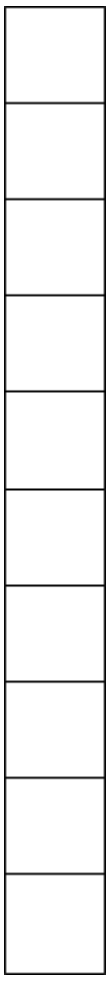
   ii. What does it signify?

Name: _____ NetID:_____

14. We know that a hash function typically keys from some domain $\mathcal{D}$ to some range $\mathcal{R}$ (typically some integer range from zero to some positive value). Suppose I make a horrible mistake in my hash function and it returns 0 for every key!

Would your hash table work? Will it work well? Give me some details about what would happen.

15. Here's a hash table of size 10. The hash function $h(x)$ is just `x modulo 10`. I'll resolve hash collisions using **quadratic hashing** with `c1=1, c2=1`. Please insert the values `[36,95,15,56,38,96,25]` into the table

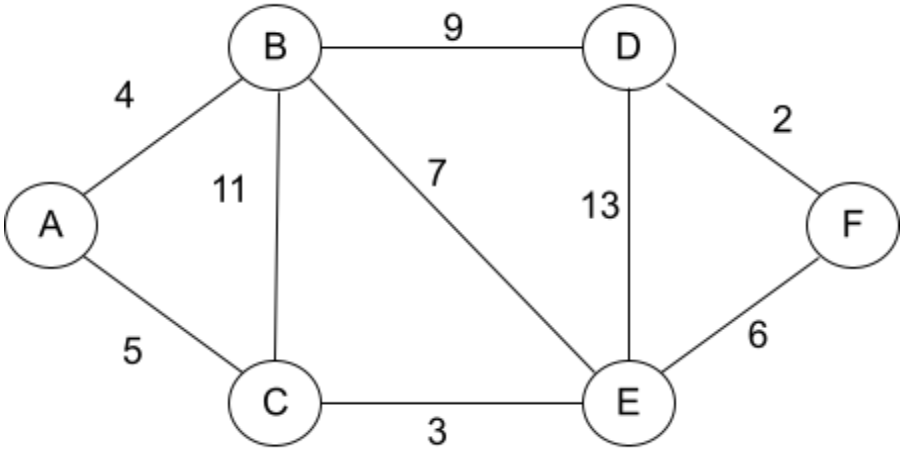


Name: _____ NetID:_____

16. Here's an undirected graph (you can think of it as a directed graph with arrows in both directions on each edge if you want). Please use Dijkstra's algorithm to compute the shortest paths from **node A to each node in the graph (show your work)**.

a. Use the provided table to neatly fill in your answers.
b. Suppose my start node was E. Would the shortest distance between E and A be the same as the value we found for A to E?

c. Is this always true? Why or why not.



Node	Shortest Distance	Prev