Write your name in box below and NetID to the right ===>	netid
	8am Lecture ☑
	12:30pm Lecture ☐
Your name goes here	

I wish you all good skill! Read these instructions FIRST!

Please put your name and NetID neatly on this page and your **name** on the rest of the pages. I unstaple these to feed them into the scanner and I have (at least once!) dropped the stack of exams and had to put them back together…

Don't use a light pencil – If the scanner can't see it, I can't grade it. If unsure, ask a proctor.

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.

Be concise, but be clear and complete. If you make assumptions, write them out. If you need extra space, there are some "scratch" pages at the end of the exam. Please tell me to look there for a continuation of your answer.

Code quality rules still apply (except for comments and rule of three). So your code must be reasonably performant (e.g. O(1) where it should be O(1)), not overly complex, use good names, **const**, etc…

For all code, you may assume that we are `using namespace std;` and that we've included all the expected files (<iostream>, <stdexcept>, <string>, <vector>, etc…)

Show your work, it helps me give you partial credit even if you get lost. I really hate taking off all the points and writing "No attempt made"

You will have until the posted end time to complete the exam.

Desks must be clear (well you can put a stuffed animal there!). No talking. No notes or electronic devices will be allowed other than a calculator (you shouldn't need one). Don't take pictures of the exam.

If you need to use the restroom, just go – no need to ask. We're adults and we should operate from a basic level of trust.

In that light, however, if you are found talking; I will move you to the front row of the classroom and you will not be allowed to continue your exam if you leave the room for any reason. No warnings, no second chances. Similarly, if I see notes or your phone or other direct indications of cheating, I will immediately terminate and confiscate your exam and grade it as is.

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.

Name: _____

1. [Syllabus] Mark **all** items that are correct regarding my late work policies
[ ] I lose 2% of my grade per day late
[ ] there is ample extra credit available to make up missing work assignments
[ ] Programming and lab assignments are sometimes extended
[ ] Participation and Challenge assignments are routinely extended

2. What is the effect of the following code? What things might happen?

```
{
  int* p = new int(33);
  delete p;
  cout << (*p) << endl;
}
```

3) I've created a circular linked list using the node structure below. Finish writing this function to print out all the values in my circular linked list to `cout` with one space between keys and a newline at the end.

```
struct Node { int key; Node* next; };
void print_circular(const Node* head) {
```

4) Why do we typically use a dummy head and a dummy tail value when we implement a doubly-linked list?

5) Write me a valid prototype for a function (not the implementation, the part I would put into a header file). It will open and read in some integers from a file that I name (an input parameter). When it's done, I will want it to provide a vector of all the even values and a count of all the other values.

Name: _____

6. I have some files that I expect to have integers in them. I want to classify them as having an even number **(type A)** or odd number **(type B)**. But I know that there might be problems. So, if I find a non-integer value, I say the file is poisoned **(type P)**. Or the file might be empty **(type E)**. Or the file is missing and doesn't even exist **(type M**). So, I want to fill in the returns below with the right classification!

But, when I did a closer code review with our grader Aditi, she said some of the returns will never, EVER happen! (We'll call those the impossible files **(type I)**).

So, bubble in the right type to return for each return statement in the code below.

```
string file_classifier(const string& filename) {
  ifstream input(filename);
  if (not input.is_open()) return   A B P E M I ;
  if (input.eof()) return           A B P E M I ;

  int count = 0;
  int x;
  while( input >> x ) {
    if (not input.good()) return    A B P E M I ;
    count++;
  }

  if (input.good()) {
    return                          A B P E M I ;
  } else {
    // Did I get an even number?
    if (count % 2 == 0) {
      if (input.eof()) {
        if (count == 0) {
          return                    A B P E M I ;
        } else {
          return                    A B P E M I ;
        }
      }
      return                        A B P E M I ;
    }

    // Must be an odd number
    else {
      if (input.eof()) {
        return                      A B P E M I ;
      } else {
        return                      A B P E M I ;
      }
    }
  }
  return                            A B P E M I ;
}
```

Type A = Even number of ints, no poison
"11 22 33 44" "55 66" (but not empty)

Type B = Odd number of ints, no poison
"11" "77 88 99" "55 44 33"

Type P = Poisoned by a non-int
"11 22 oops" "my bad" "44 five five"

Type E = An empty file (which is NOT type A!)

Type M = The file doesn't exist

Type I = The return statements I can remove because they will never execute

Name: _____

7. Please finish this definition for a Queue. Implement the expected functions (e.g. push, pop, front, etc..). If you push when the queue is full, I want it to resize and double the storage. Please implement the required destructor, I've taken care of the rule-of-three. Neither change nor add to my private member **variables**. Your implementation should have the right big O.

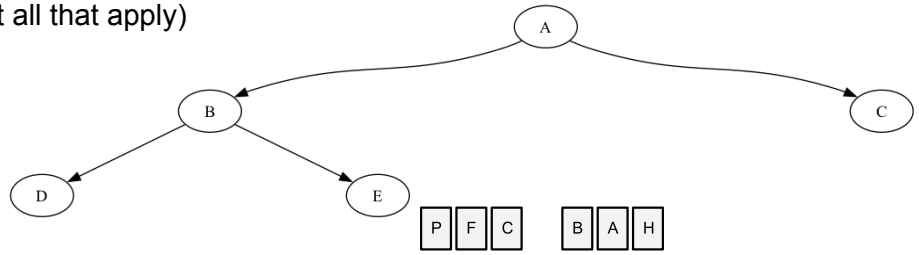
```
class Queue {
   int maxsize;
   int first;
   int size;
   string *array;
public:
  Queue() : maxsize(10),first(0),size(0),array(new string[maxsize]) {}
  Queue(const Queue&) = delete;
  Queue& operator=(const Queue&) = delete;
```

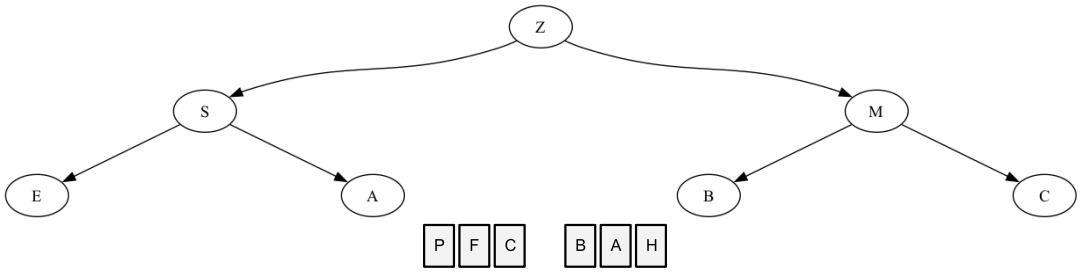
Name: _____

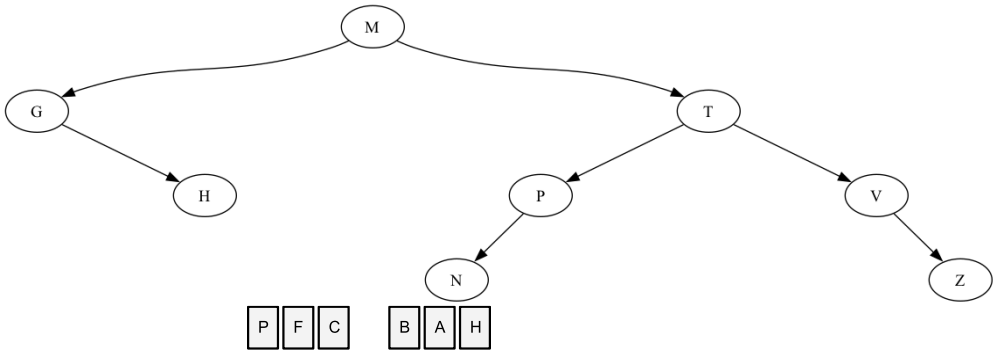
<more queue code>

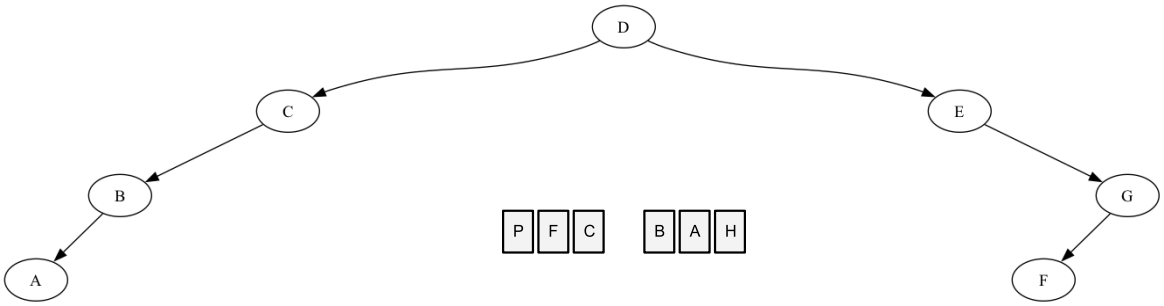
Name: _____

8. Here are some binary trees of various types. I want you to classify them as Perfect [P], Full [F], complete [C] if they qualify. Also, pick if they qualify as a BST [B], AVL [A], or MaxHeap [H] (select all that apply)









Name: _____

9. I want you to build and draw a Binary Search Tree (**BST**) by inserting the following values in order: [10, 20, 30, 40, 50, 45, 48]

10. I want you to build and draw a Adelson-Velsky-Landis Tree (**AVL** tree) by inserting the following values in order: [10, 20, 30, 40, 50, 45, 48]

Name: _____

11. In trees and their associated data structures (BST,B-Tree, AVL, Red-Black) some functions I can write iteratively or recursively (like search and insert). Others I can only write recursively. (a) Name an operation that REQUIRES a recursive approach and (b) Tell me why it CANNOT be written without recursion (or its equivalent).

12. In the queue you wrote a couple pages back, you implemented Queue::push with resize. Tell me what the big O is and justify your response.

Name: _____

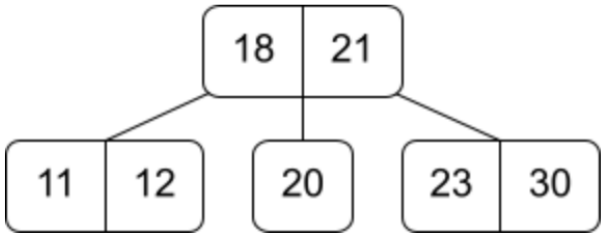
13. Here's the code for a max heap. I'll want you to fill in the code for the unimplemented functions (not the prototypes). After you get it written… tell me what it outputs in this box

Write the output here

```
#include <iostream>
#include <vector>
using namespace std;

void percolate_up(int i, vector<int>& array) {









}

// Don't write these three
void percolate_down(int i, vector<int>& array,int size);
void heapify(vector<int>& v);
void heapsort(vector<int>& v);

void heapinsert(int x, vector<int>& v) {









}

int main() {
  vector<int> e;
  heapinsert(3,e); heapinsert(7,e); heapinsert(2,e); heapinsert(10,e);
  for(const auto& x:e) { cout << x << endl; }
  return 0;
}
```

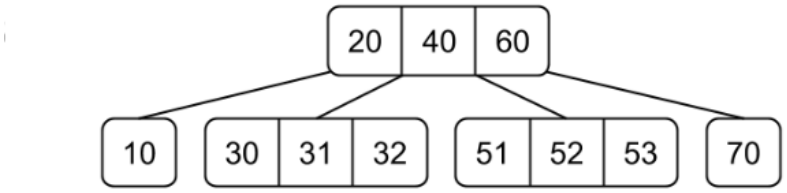
Name: _____

14. Here's a 2-3 tree. I would like you to insert the values 40, 10, and 5. Please draw the tree that results from those inserts after each step (draw three trees).



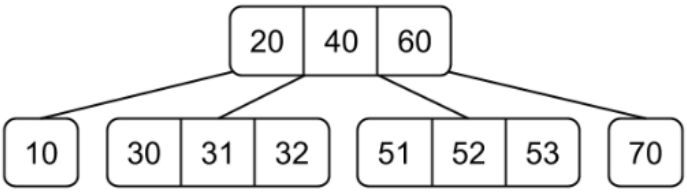
Name: _____

15. Here's a 2-3-4 tree. I would like you to insert the values 5 and 25. Draw the resulting tree after each step (so, two trees).



Name: _____

16. Here's that same 2-3-4 tree. Please delete the values 10, 20, and 30 in the standard way. Draw the tree you get after each step (3 trees)



Name: _____

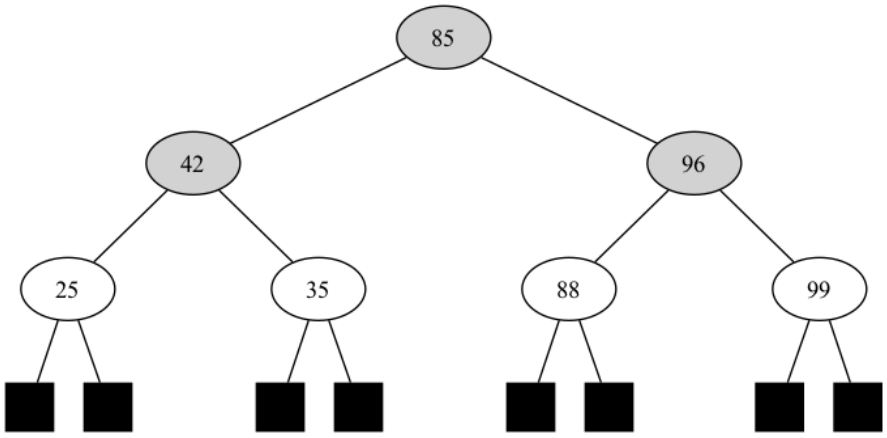
17. What is the definition of (rules for)...
    a) An AVL tree

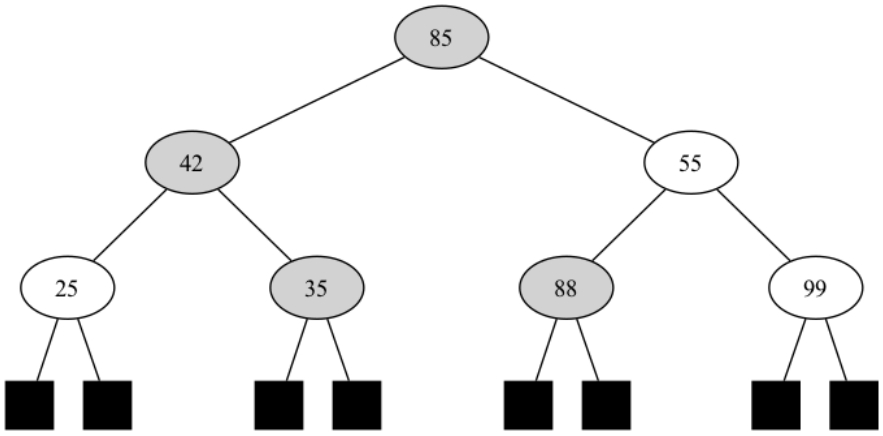
    b) A Heap

    c) A Red-Black tree

Name: _____

18. For each tree below. (a) Tell me if it is a valid Red-Black tree. (b) If not valid, list **ALL** the reasons it is not a good Red-Black tree (remember your rules from 16(b) on the last page!). Use the node numbers in your reasons (e.g. Node 11 is the root and it isn't black).





Name: _____

19. I would like you to explain bubble sort to me. Imagine that we are both looking at the source code (you do not need to write it here). I want to know what is going on, I want to know the Big O (in relation to *n*, the size of the array), AND I want to know why you think it is that big O.

20. Your Linux sysadmin cannot recover your password if you lose it because the password is not saved on the system. But if the password isn't there, how does logging in work (assuming you've typed in a correct username and password)?

Name: _____

21. For each of the following, tell me the Big O and why you choose that value. You will need to describe each of the significant parts of the algorithm in sufficient detail to account for the work. Detail what you are counting. Don't be vague (like only saying "height of the tree!"). Relate the action to the complexity of the work (remember what Big O tells us!) Again, imagine that we are both looking at the code [again, don't write it!] and you are explaining it to me.

a) Mergesort

b) Linked list reverse

c) Queue pop (Linked list)

d) Heap/Priority Queue remove

Name: _____

22. I tell you the three things you need to make a hash table
- The hash function is modulo 10 (so the last decimal digit of a number)
- The slots are either blank (no value) or a number [we aren't deleting]
- My collision strategy is quadratic probing to find an empty slot (with the standard c_1 and c_2 both set to 1)

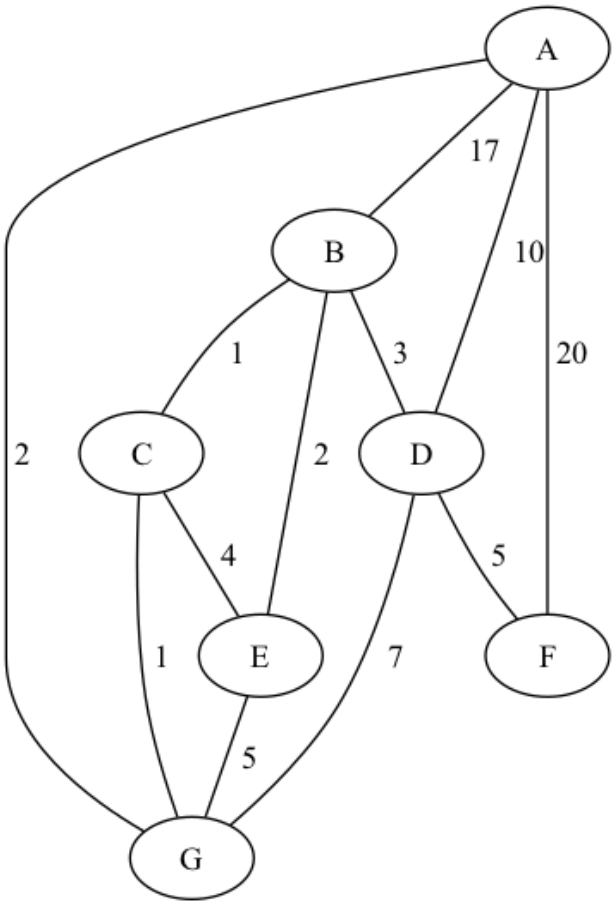
Please insert the values [33, 47, 55, 57, 20, 30, 90, 99] into the hash table so they land in the appropriate slot.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Name: _____

23. Just write out the steps you will take to implement Dijkstra's Method. Once you have the steps, you'll use them for the graph below. Keep careful track of your work since I want to see each step (so cross out old work). Each node needs a final distance from the start node "**A**", and all of the "prev" nodes you found in the backwards path.

24. Fill in this table using the Dijkstra steps, So, mark each node visited as you visit it. I need to see **all** your work, so as you update things (e.g. the shortest distance), CROSS OUT THE OLD ONE and then write in the new one.



Label	V	Shortest distance found so far	Previous node in backward path
A			
B			
C			
D			
E			
F			
G			

Name: _____

Scratch 1

Name: _____

Scratch 2