# **Exam 1**

*This is a closed book and closed notes test.* You are not allowed to have anything on your desk other than pencil and this exam paper during the test; this includes *calculators* or *electronic assistance* of any kind – ***especially smartphones***.

*You may not leave to go to the restroom.* Please go before the exam starts.

*You may not ask questions.* If something is confusing, write a note beside the question and explain your assumptions.

*You must show all of your work on this exam.* You will not be allowed to turn in additional sheets of paper.

*Read and sign the following statement.*  Failure to sign the statement will result in a **zero** on the exam.

*I have neither given nor received unauthorized assistance on this test. I have notified the proctor of any violations of the above policies.*

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **Problem** | **Score** |
| 1 | / 30 |
| 2 | / 30 |
| 3 | / 20 |
| 4 | / 20 |
| **Total** | **/ 100** |

*Points divided evenly among parts of a problem unless otherwise specified.*

1. (30 points) Evaluate the following C++ code snippets.

class Pizza {

private:

int diameter;

int slices;

public:

bool large;

// approximate pi as 3

double area() { return (diameter\*diameter/4)\*3; }

public:

Pizza(int d=12,

int s=8) {

diameter = d;

slices = s;

if (diameter > 14) large = true;

else large = false;

}

int getSlices() { return slices; }

int getDiameter() { return diameter; }

double areaPerSlice() {

return area() / slices;

}

};

Consider these variables declared within a function:

Pizza large(16);

Pizza personal(6, 4);

Pizza medium;

Pizza small(10.0,6);

Evaluate the following expressions. Or, if the expression is illegal, explain why.

|  |  |  |
| --- | --- | --- |
| **#** | **Expression** | **Evaluate or if illegal write “Syntax error” and explain why** |
| 1 | large.getDiameter() |  |
| 2 | medium.getSlices() |  |
| 3 | large.areaPerSlice() > small.areaPerSlice() |  |
| 4 | personal.getSlices() |  |
| 5 | medium.areaPerSlice() |  |
| 6 | medium.large |  |
| 7 | large.large |  |
| 8 | large.areaPerSlice() |  |
| 9 | medium.medium |  |
| 10 | personal.area() |  |

2. (30 points) Design a class to implement a Queue of integers. Your class should support push, pop, peek, and empty. You must use a linked list as your implementation of the queue, and may not assume some other code implements the linked list. Your code must include that implementation.

(left empty for problem 2)

3. (20 points) Consider the following operations on a singly linked list of length N. State the big-O complexity of each operation and justify your answer.

1. Insert a new node before a node that you have a pointer to.
2. Append an item to the end of a list given a pointer to the tail of the list.
3. Insert an item in the middle of the list, given the head pointer.
4. Print the nodes of the linked list in reverse.

4. (20 points) Give an example of C++ code for each concept below and briefly (in one comment/sentence) explain why it demonstrates the concept. You do not need to show a full implementation. Use comments to fill in gaps to avoid writing a lot of code.

1. Abstract Base Class.
2. A default class constructor that takes a parameter.
3. A private class member.
4. Is-A relationship between classes.

**C++ Keywords**

In common with C:

auto const double float int short struct unsigned  
break continue else for long signed switch void  
case default enum goto register sizeof typedef volatile  
char do extern if return static union while

Unique to C++:

asm dynamic\_cast namespace reinterpret\_cast try  
bool explicit new static\_cast typeid  
catch false operator template typename  
class friend private this using  
const\_cast inline public throw virtual  
delete mutable protected true wchar\_t

Reserved words:

and bitand compl not\_eq or\_eq xor\_eq  
and\_eq bitor not or xor

**ASCII Table**

