**CS 2420 2021 Fall Midterm 1**

1 **(5 points)**. What are the run times of each of the following for-loops? Please answer the question using Big-O notation.

for ( int i = 0; i < n; ++i ) {

for ( int j = 0; j < m; ++j ) {

sum += i + j;

}

}

for ( int i = 0; i < n; ++i ) {

for ( int j = 0; j < n\*n; ++j ) {

sum += i + j;

}

}

for ( int i = 0; i < n1; ++i ) {

for ( int j = 0; j < n2; ++j ) {

for ( int k = 0; k < n3; ++k ) {

sum += i + j + k;

}

}

}

2 **(5 points)**. Why do the following two for-loops have the same run time complexity?

for ( int i = 1; i <= n; i \*= 2 ) {

sum += i;

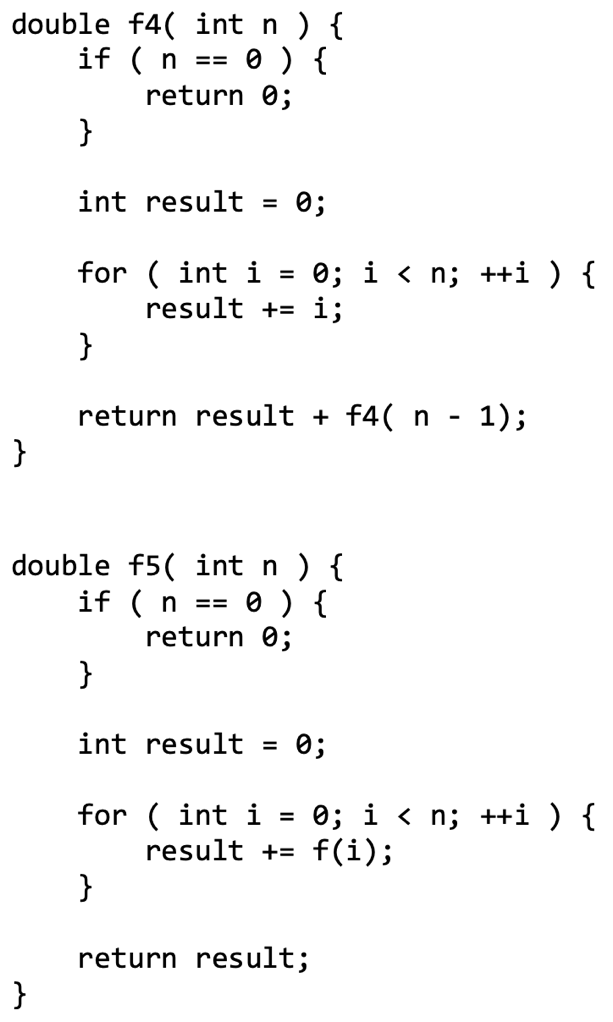
}

for ( int i = 1; i <= n\*n; i \*= 2 ) {

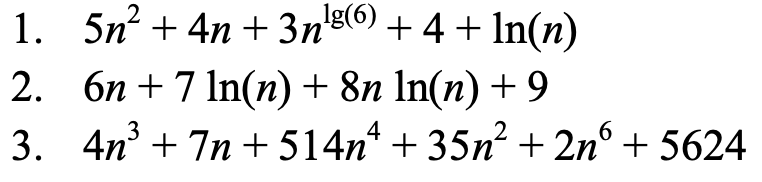
sum += i;

}

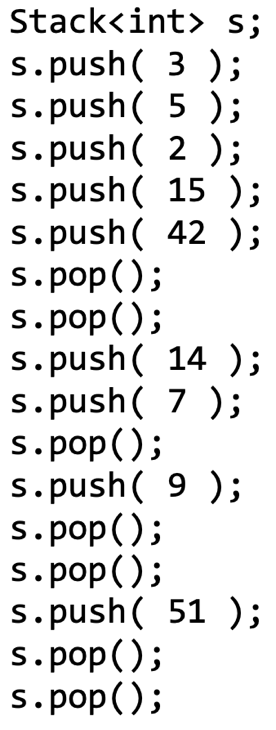
3 **(5 points)**. Write down the runtime complexity of the following function:



4 **(5 points)**. Find the most appropriate representative element that describes each of the following rates of growth. For example, the most appropriate representative of 3n2 + 4n ln(n) + 5n + 2 is n2



5 **(5 points)**. What is the state of the stack after the following sequence of pushes and pops? You may draw a stack to show the progress of each operation.



6 **(5 points)**. Suppose the numbers 0, 1, 2, …, 9 were pushed onto a stack in that order, but that pops occurred at random points between the various pushes. The following is a valid sequence in which the values in the stack could have been popped:

3, 2, 6, 5, 7, 4, 1, 0, 9, 8

Explain why it is not possible that

3, 2, 6, 4, 7, 5, 1, 0, 9, 8

is a valid sequence in which the values could have been popped off the stack.

7 **(5 points)**. Give three applications in your daily life that can be implemented (or explained) with stack and queue.

Stack:

Queue:

8 **(5 points)**. Suppose we wanted to implement a member function clear() which removed all the current entries from a stack class that uses an array internally. How would you implement such a function?

9 **(10 points)**. When we use an internal array to store the values of a stack, one strategy for when the array is full is to double the size of the array. This may lead to a very large array which may not always be necessary. One strategy would be to halve the size of the array if the capacity of the array drops below a certain value. Explain why it would be a bad idea to halve the size of the array if the array is half empty. Justify your answer using algorithm analysis we have discussed in the lecture.

10 **(5 points)**. Which of the following are valid and complete reverse Polish expressions? The operators + and \* represent the standard binary addition and multiplication operations, respectively.

1 2 + \* 4 5 \* 6 + +

1 2 3 + \* 4 5 \* + +

1 2 + 3 \* 4 5 \* 6 +

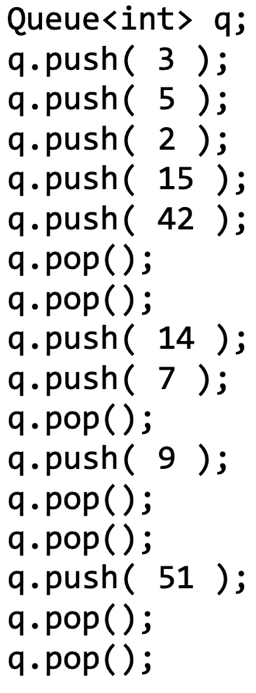
11 **(5 points)**. Evaluate the following expressions that are written using reverse Polish notation. You need to show the progress of your evaluation.

1 2 3 + \* 4 5 \* 6 + +

1 2 3 + \* 4 5 \* + 6 +

1 2 + 3 \* 4 5 \* 6 + +

12 **(5 points)**. What is the state of the queue after the following sequence of pushes and pops? You may draw a stack to show the progress of each operation.



13 **(10 points)**. A queue is a first-in—first-out data structure. Suppose you have two queues accepting requests for a particular service, but at some point, the server for one queue goes down. You must now merge the two queues. If no other information is available, what might a reasonable strategy be for forming a single queue out of the two queues? What additional information would be necessary to form a single queue in such a way to be most fair to all requests?

14 **(5 points)**. A queue is a first-in—first-out data structure. Suppose a queue is accepting requests for a particular service and that the queue is finite in size and cannot be resized. If the queue is full and another request comes in, is it fairer to replace an item in the queue or to discard the request.

15 **(20 points)**. An implementation could potentially use any of the following data structures to implement the Queue ADT. Indicate the best-case run times if we use the listed data structures in as optimal a sense as possible. Choose the implementation that minimizes the run time of as many operations as possible. In the case of arrays, you may have to move entries if the array is not full—an Θ(n) operation.

