CS 225	**Introduction to Computer Science**	Test 2
Name:		Maximum Points: 54

Important: This is an open book test. You can use any books, notes, or paper. *Do not log into the computer during the test.* Any calculations and rough work can be done on the back side of the test pages. If there is a syntax error in any program segment, just write it down and you will get full credit for the problem.

1. [6 pt] Give an example of an algorithm that is

   (a) $O(1)$

   (b) $O(N)$

   (c) $O(N^2)$

2. [6 pt] Algorithm 1 does a particular task in a "time" of N^3 where N is the number of elements processed. Algorithm 2 does the same task in a "time" of $3N + 1000$.

   (a) What are the Big-O requirements of each algorithm?

   (b) Which algorithm is more efficient by Big-O standards?

   (c) Under what conditions, if any, would the "less efficient" algorithm execute more quickly than the "more efficient" algorithm?

3. [5 pt] Explain why the cost of fixing an error is higher the later in the software cycle the error is detected.

4. [5 pt] Write the loop that is described by the following loop invariant:

   (a) $1 \leq \texttt{index} \leq \texttt{maxlist}$

   (b) 0 is not found in `list[i]`, $i = 1, 2, \ldots,$ `index-1`

   (c) $\texttt{sum} = \sum_{i=1}^{\texttt{index}-1} \texttt{list[i]}$

5. [4 pt] Differentiate between "data coverage" and "code coverage" in program testing. Which is better?

6. [5 pt] Describe the accessing function of C one-dimensional array at the logical level.

7. [5 pt] Show what is written by the following segment of code, given that `stack` is a stack of integer elements, and `x`, `y`, and `z` are integer variables.

```
stack = create_stack ( stack);

x = 1; y = 0; z = 4;

stack = push ( stack, y );
stack = push ( stack, x );
stack = push ( stack, x + z );
y = pop ( stack );
stack = push ( stack, z * z );
stack = push ( stack, y );
stack = push ( stack, 3 )p;
printf ( "x = %d\ny = %d\nz = %d\n", x, y, z );
while ( ! empty_stack ( stack ) )
{
    x = pop ( stack );
    printf ( "%d\n", x );
}
```

8. [8 pt] Assuming that data of `stack_element_type` takes 24 bytes, integer takes 4 bytes, and `max_stack` = 100, compare the space requirements of static array-based versus dynamic linked stack implementations. (In calculating the space requirements of the linked implementation, don't forget to count the external pointer).

Number of elements	Static array-based	Dynamic linked stack
0		
10		
50		
100		

9. [10 pt] Use the following definition

```
typedef struct queue_node_type
{
    queue_element_type     element;  /* The information field        */
    struct queue_node_type *next;    /* Pointer to the next element  */
};

typedef struct
{
    struct queue_node_type *front, *rear;   /* Front and rear of queue */
} queue_type;
```

Now, write a function `queue_count` with the following specifications

Function. Returns the number of elements in the queue

Input. A queue given by `queue_type queue;`

Preconditions. Queue has been created

Output. Number of elements in the queue

Postconditions. Returns the number of elements in the queue