Daniel Gaviria Rodriguez,

Technical Assessment theorical answers.

1a) Describe the difference between a stack and a queue.

A stack is a type of linear data structure in which its elements can be inserted and deleted from one side of the list. Which is called the top. Stacks follow the LIFO rule (Last In First Out), which means that the last inserted element will be the first element to come out.

A Queue on the other hand is a type of linear data structure in which the elements can only be inserted from a side called rear, and its elements can only be deleted from a side called the front. Queues follow the FIFO (First In First Out) rule, which means that the first element to be inserted, will be the first element to be removed from the queue.

1b) Describe a typical programming use for a queue.

Queues are usually used when something doesn't have to be processed immediately. But has to be processed in First In First Out order.

2a) Use C/C++ to show the operations necessary to insert a list element given a pointer to the new element and a pointer to the element after which the new element needs to be inserted.

In the **double_ node2 class**, the pointers ***next and *prev** work as pointers for the next and previous node of the list.

***B, *E, *T** pointers work as beginning and ending pointers.

***T** works as a "temporal" pointer that will take a variable **x** value which will fill a field by pointing the **id** name. ***T** will also point towards the address of ***next and *prev** and initialize them in NULL.

3a) Use C/C++ to show the structure of an element of a sorted, binary tree in which each element contains a single character. Describe briefly what any pointers point to.

Node_b stores an address for the***Left and *Right** pointers to use.

***R** is the root pointer through which information in the binary tree can be accessed other pointers will point towards the address of ***R** in order to access the information of the base node.

***f** Stores the addresses of both ***Left and *Right**. ***f** also stores the information of ***Left and *Right.**

***T** takes the data stored in the **x** variable, which is **id** and points towards ***Left or *Right** depending on the input.

4) Compare the following function and macro definitions. In what cases will they produce different results and/or side effects?

```
int square(int val) { return val*val; }
```

```
#define square(val) (val*val).
```

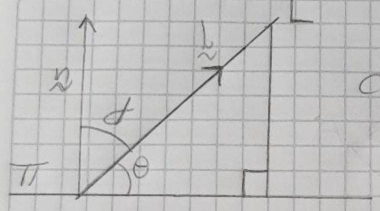
Since macros are preprocessed, there’s a higher possibility for them to cause different results. Due to them not being type checked, or are checked by the compiler directly. However, that’s not to say that macros are that bad.

That is due to the fact that macros are replaced by their value once the code is executed, thus making the execution times faster compared to that of a function. Which is a notable side effect.

6a) Given a 2D vector A in the x-y plane of length |A| and angle theta to the x-axis, give the equations for the x and y components of A.

6b) Given the x,y,z components of a 3D vector A, give the equation for the angle between the vector and the x-y plane.

DD MM AA



$$\cos\alpha = \left| \frac{\underset{\sim}{l} \cdot \underset{\sim}{n}}{|\underset{\sim}{l}|\,|\underset{\sim}{n}|} \right| = \frac{\pi}{2} - \alpha$$

7a)