C PROGRAMMING HOMEWORK

S.ABHISHEK

**1.SEGMENTATION FAULT**

Segmentation faults are a common class of error in programs written in languages like [C](https://en.wikipedia.org/wiki/C_(programming_language)) that provide low-level memory access.

They arise primarily due to errors in use of [pointers](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) for [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) addressing, particularly illegal access.

A segmentation fault occurs when a program attempts to access a [memory](https://en.wikipedia.org/wiki/Computer_memory) location that it is not allowed to access, or attempts to access a memory location in a way that is not allowed (attempting to write to a [read-only](https://en.wikipedia.org/wiki/Read-only_memory) location).

**EXAMPLE**

char \*str = "Foo"; // Compiler marks the constant string as read-only

\*str = 'b'; // Which means this is illegal and results in a segmentation fault.

**2.DIFFERENCE BETWEEN VARIABLE LENGTH AND DYNAMIC MEMORY ALLOCATION**

**(1) VLA**

**(2) DYNAMIC MEMORY ALLOCATION**

The major difference, in terms of usability and behaviour are:

1. (1) is on stack, usually while (2) is on heap, always.
2. (1) has fixed size once allocated, (2) can be resized.
3. (1) is allocated when the enclosing function is called and has the block scope, (2) is allocated memory dynamically, at runtime and the returned memory has a lifetime which extends from the allocation until the deallocation.
4. (1) allocated memory need not be managed by programmer, while in (2) all malloc () memory should be free ().

The VLA array will only be valid inside the scope where it is declared, while a dynamic array will be available everywhere in the program until you call free ().

In practice, VLAs may be faster than dynamic memory, in case the compiler uses stack allocation for the VLA.

In DYNAMIC MEMORY ALLOCATION memory is more explicitly (but more flexibly) managed, typically by allocating it from the "heap", an area of memory structured for this purpose.

In C, the library function malloc () is used to allocate a block of memory on the heap.

The program accesses this block of memory via a POINTER that MALLOC () returns.

When the memory is no longer needed, the pointer is passed to free which deallocates the memory so that it can be used for other purposes.

**3. DANGLING POINTER, NULL POINTER, VOID POINTER AND WILD POINTER**

**DANGLING POINTER –**

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer.

**NULL POINTER –**

NULL Pointer is a pointer which is pointing to nothing.

In case, if we don’t have address to be assigned to a pointer, then we can simply use NULL.

**VOID POINTER -**

Void pointer is a specific pointer type – void \* – a pointer that points to some data location in storage, which doesn’t have any specific type.

Void refers to the type. Basically, the type of data that it points to is can be any.

If we assign address of char data type to void pointer it will become char Pointer, if int data type then int pointer and so on.

Any pointer type is convertible to a void pointer hence it can point to any value.

Void pointers **cannot be dereferenced**. It can however be done using typecasting the void pointer.

Pointer arithmetic is not possible on pointers of void due to lack of concrete value and thus size.

**WILD POINTER –**

A pointer which has not been initialized to anything (not even NULL) is known as wild pointer.

The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

!... THANKYOU ...!