CSCI3240 Lab 8:

Dynamic Memory Allocation

Dynamic memory allocation is necessary to manage available memory. For example, during compile time, we may not know the exact memory needs to run the program. So, for the most part, memory allocation decisions are made during the run time. C also does not have automatic garbage collection like Java does. Therefore, a C programmer must manage all dynamic memory used during the program execution. The provides four functions that can be used to manage dynamic memory.

These functions can be found in the <stdlib.h> header file.

Function	Description
malloc	Allocates a block of memory but doesn't initialize it.
calloc	Allocates a block of memory and clears it (makes it 0).
realloc	Resizes a previously allocated block of memory.

Of the three, `malloc` is probably the most used. It's more efficient than `calloc`, since it doesn't have to clear the memory block that it allocates.

When we call a memory allocation function (malloc, `calloc` or `realloc`) to request block of memory, the function has no idea what type of data we're planning to store in the block, so they return a value of type **`void *`**. A **`void *`** value is a "generic" pointer—essentially, just a memory address.

Null Pointers

When we call a memory allocation function, there's allocation function, there's always a possibility that it won't be able to locate a block of memory large enough to satisfy our request. If that happen, the function will return a null pointer. A null pointer is pointer to nothing. After we've stored the return value of memory allocation function in a pointer variable (say P), we must test P to see if its's a null pointer.

The null pointer is represented by a macro named NULL, so we can test malloc's return value in the following way:

```
int *P = malloc(500* sizeof(int));
if (P == NULL){
    printf("Memory Allocation failed!!");
    exit(0);
}
// Do something
free();  //release the memory
```

Malloc Function:

The “malloc” or “memory allocation” method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form. It doesn’t Initialize memory at execution time so that it has initialized each block with the default garbage value initially.

Syntax:

```
cast-type *ptr;
ptr = (cast-type*) malloc(byte-size);
```

For Example:

```
ptr = (int*) malloc(100 * sizeof(int));
```

This statement will allocate 400 bytes of memory. And, the pointer ptr holds the address of the first byte in the allocated memory. You can then use ptr as an array.

```
ptr[50] = 33;                        //stores 33 in index location 50 of ptr.
int sum = ptr[0] +ptr[2];  // stores sum of values stores in index 0 and 1 of ptr.
```

Calloc Function:

“calloc” or “contiguous allocation” method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. it is very much similar to malloc() but has two different points and these are:

- It initializes each block with a default value ‘0’.
- It has two parameters or arguments as compare to malloc().

Syntax:

```
ptr = (cast-type*)calloc(n, element-size);
//here, n is the no. of elements and element-size is the size of each element.
```

For Example:

```
ptr = (float*) calloc(25 , sizeof(float));
```

This statement allocates contiguous space in memory for 25 elements each with the size of the float.

Realloc Function:

The “realloc” or “re-allocation” method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to dynamically re-allocate memory. Re-allocation of memory maintains the already present value, and new blocks will be initialized with the default garbage value.

Syntax:

```
ptr = realloc(ptr, newSize);
//where ptr is reallocated with new size 'newSize'.
```

If space is insufficient, allocation fails and returns a NULL pointer.

Free Function:

The “free” method in C is used to dynamically de-allocate the memory. The memory allocated using functions malloc() and calloc() is not de-allocated on its own. Hence the free() method is used, whenever the dynamic memory allocation takes place. Memory leaks occur if you forget to free the dynamically allocated memory. Over time, such leaks can exhaust available system memory.

Syntax:

```
free(ptr);
```

Next, please finish the **Lab8_DynamicMemoryAllocation** quiz in D2L.

References:

https://www.tutorialspoint.com/what-is-dynamic-memory-allocation-in-c

https://www.tutorialspoint.com/c_standard_library/c_function_malloc.htm

https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/

https://www.cs.cmu.edu/~guna/15-123S11/Lectures/Lecture08.pdf