Homework 1: Research

By: Adriana Rivera

These functions are for allocate block of memory.

We allocate memory in order to store data there

The Malloc function

"The **malloc()** function stands for the memory size of a determined pointer. It is a function which is used to allocate a block of memory dynamically. It reserves memory space of specified size and returns the null pointer pointing to the memory location. The pointer returned is usually of type void. It means that we can assign malloc function to any pointer."

void* malloc(size_t size);

*void*p = malloc(number_of_elements * sizeof(int));*

size_t= stores integer positive values

All of this is in bytes.

The malloc function returns a void pointer (a number) that gives us the address of the first byte in this block (of memory that it allocates)

Example:

```
#include <stdlib.h>
int main(){
int *ptr;
ptr = malloc(15 * sizeof(*ptr)); /* a block of 15 integers */
    if (ptr != NULL) {
      *(ptr + 5) = 480; /* assign 480 to sixth integer */
      printf("Value of the 6th integer is %d",*(ptr + 5));
    }
}
```

The Calloc function

"The **calloc()** function in C is used to allocate a specified amount of memory and then initialize it to zero. The function returns a void pointer to this memory location, which can then be cast to the desired type. The function takes in two parameters that collectively specify the amount of memory to be allocated.

Calloc takes two arguments:

Void* calloc(size_t num, size_t size);

- First: Number of elements of a data type

- Second: Size of the data type

*void*p = (int*) calloc(number_of_elements, sizeof(int));*

```
#include<stdio.h>
#include<stdlib.h>
int main() {
    int* a = (int*) calloc(5, sizeof(int));
    return 0;
```

It sets all byte positions to 0.

The Realloc function

“In the C Programming Language, the realloc function is used to resize a block of memory that was previously allocated. The realloc function allocates a block of memory (which be can make it larger or smaller in size than the original) and copies the contents of the old block to the new block of memory, if necessary.”

void * realloc(void *ptr, size_t size);

This function changes the size of the block of memory.

two arguments

- Pointer of starting address
- Size new block

This can create a new block and copy the previous data whatever bytes were written in the previous block.

The realloc function returns a pointer to the beginning of the block of memory. If the block of memory cannot be allocated, the realloc function will return a null pointer.

```
#include <stdio.h>
#include <stdlib.h>

int main () {
   char *str;

   /* Initial memory allocation */
   str = (char *) malloc(15);
   strcpy(str, "tutorialspoint");
   printf("String = %s,  Address = %u\n", str, str);

   /* Reallocating memory */
   str = (char *) realloc(str, 25);
   strcat(str, ".com");
   printf("String = %s,  Address = %u\n", str, str);

   free(str);

   return(0);
}
```

The Free function

When a block of memory allocated by malloc is no longer used, we can deallocate it using the free function.

Void free (void* ptr);

```
#include <stdlib.h>

int* ptr;/* pointers to integer */
int* ptr2;
...
ptr = (int*)malloc ( 300*sizeof(int) );
...
  ptr[33] = 15;             /* works memory area */

  full_of_zeros (10,ptr);   /* example */

  ptr2 = ptr + 15;          /* assigned to another pointer */

/* finally, we free the memory block */
free(ptr);
```

References:

- https://www.guru99.com/malloc-in-c-example.html
- https://www.youtube.com/watch?v=xDVC3wKjS64
- https://www.educative.io/edpresso/what-is-calloc-in-c
- https://www.techonthenet.com/c_language/standard_library_functions/stdlib_h/realloc.php
-