Homework 1

Brayan Fonseca Carrillo

Robotics 2°A

Unit 2

Structure programming

Luis Gerardo Camara Salinas

Due date: 1/07/21

Malloc ()

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.

Syntax

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

Example

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

Realloc ()

Realloc() is a function of C library for adding more memory size to already allocated memory blocks. The purpose of realloc in C is to expand current memory blocks while leaving the original content as it is. realloc() function helps to reduce the size of previously allocated memory by malloc or calloc functions. realloc stands for reallocation of memory.

Example

```
#include <stdio.h>

int main () {

   char * ptr;

   ptr = (char *) malloc (10);

   strcpy (ptr, "Programación");

   printf ("% s, Dirección =% u \ n", ptr, ptr);

   ptr = (char *) realloc (ptr, 20); // ptr se reasigna con un nuevo tamaño

   strcat (ptr, "En 'C'");

   printf ("% s, Dirección =% u \ n", ptr, ptr);

   libre (ptr);

   return 0;

}
```

Calloc ()

The C library function **void *calloc(size_t nitems, size_t size)** allocates the requested memory and returns a pointer to it. The difference in **malloc** and **calloc** is that malloc does not set the memory to zero where as calloc sets allocated memory to zero.

Syntax

void * calloc (size_t nitems, size_t size)

Example

```
#include <stdio.h>

#include <stdlib.h>

int main () {

   int i, n;

   int * a;

     printf ("Number of elements to be entered:");

   scanf ("% d", & n);

      a = (int *) calloc (n, sizeof (int));

   printf ("Enter% d numbers: \ n", n);

   for (i = 0; i <n; i ++) {

      scanf ("% d", & a [i]);

   }

   printf ("The numbers entered are:");

   for (i = 0; i <n; i ++) {

      printf ("% d", a [i]);

   }

   free (a);

   return (0);

}
```

free ()

The **free() function** in C library allows you to release or deallocate the memory blocks which are previously allocated by calloc(), malloc() or realloc() functions. It frees up the memory blocks and returns the memory to heap. It helps freeing the memory in your program which will be available for later use.

In C, the memory for variables is automatically deallocated at compile time. For dynamic memory allocation in C, you have to deallocate the memory explicitly. If not done, you may encounter out of memory error.

Syntax

```
void free (void * ptr)
```

Example

```
#include <stdio.h>

int main () {

int * ptr = malloc (10 * sizeof (* ptr));

if (ptr! = NULL) {

   * (ptr + 2) = 50;

   printf ("Value of the 2nd integer is% d", * (ptr + 2));

}

free (ptr);

}
```