Lab 8 - Exercise – All sections
Arrays of Dynamic Memory & Pointers
CS 262 – Spring 2022

The purpose of this activity is to practice how ask Dynamic Memory to create an array and *use pointers to access the elements of it*. The functions in C that can be used to allocate dynamic memory are `malloc()` and `calloc()`.

```
void *malloc(size_t size);
```

- The `malloc()` function reserves a contiguous memory block, **`size`** is the memory block in **bytes**. When a program obtains a memory block through `malloc()`, its contents are undetermined.
- This function returns a pointer to the allocated memory, or **`NULL`** if the request fails

```
void *calloc(size_t num, size_t size);
```

- The `calloc()` function reserves a block of memory for an array of **`num`** elements, each element is **`size`** **bytes** long. When a program obtains a memory block through `calloc()`, its contents are bits to zero.
- This function returns a pointer to the allocated memory, or **`NULL`** if the request fails

Note that the type of the return pointer for both functions is always **`void*`**, which can be cast to the desired type of data pointer in order to be dereferenceable.

Example:

Use dynamic memory to create an array of `int` of size `N`.
Here is an example how we could implement `malloc()` and `calloc()` to create the array.

```
int N;
int *N1, *N2
// Ask the user to input the number of elements for the array (N)

//malloc gets ONE parameter which indicates the TOTAL of bytes
      N1 = (int*) malloc( N*sizeof(int));

//calloc gets TWO parameters: number of elements, size in bytes of ONE single element
      N2 = (int*) calloc( N,sizeof(int));

//Use N1 and N2 as usually you do with an array defined with [ ], here an example:
      N1[0] = 100;
      N2[0] = 200;

//Do not forget to free the memory before the program ends
      free(N1);
      free(N2);
```

Note that in this example there is no validation in case the malloc() or calloc() returns NULL. Make sure to use an if-condition to check the return value; otherwise, you will get a segmentation error in case the pointer is NULL and you are trying to access the memory location.

Description of the program

Code a program that creates a dynamic memory array of integers, fills the array with random numbers, and prints the contents of the array using pointers to access each element.

General steps:

- Prompts the user to enter an input number (`N`), where `N>0`
- Use `malloc()` or `calloc()` to create an array `items` with `N` elements.
- Declare a pointer `*p` to points the array `items`.
- In a loop fill each element of the array with a random number `[1-N]` inclusive.
- Print out the elements of `items` using the pointer `*p` instead of `items[index]`.
- Print each element in the same line separated by a single space
- Free the *dynamic memory* at the end of the program

Requirements:

- Use `fgets()` and `sscanf()` to get the `N` value
- Use `calloc()` or `malloc()` to ask for *dynamic memory* for the array
- Use `srand()` and `rand()` to generate the random numbers
- Use `srand(time(NULL));` to randomize the seed
- Include the library `time.h` to have access to the function `time()`

Example 1:

```
Enter the number of elements of the array: 3
Content of the array:
3 2 2
```

Example 2:

```
Enter the number of elements of the array: 5
Content of the array:
5 4 4 1 2
```

Example 3:

```
Enter the number of elements of the array: 7
Content of the array:
1 5 6 1 7 3 2
```