Ramzy Izza Wardhana
21/472698/PA/20322
IUP CS B

Activity 10.3 – Memory Management

Compile the changes that you made and run the program.

free.c program. Add free(ptr); before the for loops

```
 GNU nano 6.2                                                  free.c *
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
int main(){
        int *ptr;
        ptr = (int *) malloc (5 * sizeof(int));
        if(ptr != NULL){
                printf("Memory has been successfully allocated.\n");
                printf("Starting address: %p\n", ptr);
                printf("End address: %p\n", ptr+4);
                free(ptr);
                for(int i=0;i<5;i++){
                        ptr[i] = i+1;
                }
                printf("The elements of the array are:\n");
                for(int i=0;i<5;i++){
                        printf("%d\n",ptr[i]);
                }
        }
}
```

1. What is the result?

```
root@--:~# ./free.out
Memory has been successfully allocated.
Starting address: 0x55ba653ab2a0
End address: 0x55ba653ab2b0
The elements of the array are:
1
2
3
4
5
```

   Answer: The result is elements inside the array starting from 1 – 5.

2. Can the pointer still be used to store and print values? Why
   Answer: The pointer can still be used to store and print value as long as there are still available space on it. In other words, if the memory space/slot has already been used by other processed, we cannot store and print the value, but in my case above, since we have free spaces to allocate our data, therefore all the data will be print from 1 – 5.