AROR UNIVERSITY OF ART, ARCHITECTURE, DESIGN & HERITAGE SUKKUR

Operating System
Lab-06

Compiled by: Aurangzeb Magsi

Course Title: OperatingSystem

4. Free()

Lab-5

Dynamic Memory Allocation and Pointers using C

1. Malloc()
2. Calloc()
3. Realloc()

Dynamic Memory Allocation using C

1. Dynamic Memory Allocation

Dynamic memory allocation refers to managing system memory at runtime. Dynamic memory management in C programming language is performed via a group four functions named malloc(), calloc(), realloc(), and free(). These four dynamic memory allocation functions of the C programming language are defined in the C standard library header file . Dynamic memory allocation uses the heap space of the system memory.

Malloc():

"malloc" or "memory allocation" is used to allocate a block of memory on the heap. Specifically, malloc() allocates the user a specified number of bytes but does not initialize. Once allocated, the program accesses this block of memory via a pointer that malloc() returns

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

Example: Memory Allocation (malloc)

```
#include <stdio.h>
#include <stdlib.h> %
int main(){
int* ptr;
int n, i;
printf("Enter number of elements:");
scanf("%d",&n);
printf("Entered number of elements: %d\n", n);
ptr = (int*)malloc(n * sizeof(int));
if (ptr == NULL) { printf("Memory not allocated.\n");
}
else {
printf("Memory successfully allocated using malloc.\n");
for (i = 0; i < n; ++i) {
 ptr[i] = i + 1;
}
printf("The elements of the array are: ");
for (i = 0; i < n; ++i) {
printf("%d ", ptr[i]);
}
}
return 0;
 }
```

LAB Task-01:

Write a C program that:

1. Dynamically allocates memory for n integers using malloc.
2. Stores the **square of each index (i*i)** in the allocated memory (i.e., 0, 1, 4, 9, 16, ...).
3. Displays the values to the user.
4. Frees the allocated memory at the end.

Expected Output:

Enter number of elements: 5
Memory successfully allocated using malloc.
The square values are: 0 1 4 9 16
Memory freed successfully.

2. calloc():

"calloc" or "contiguous allocation" method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. It initializes each block with a default value '0'.

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

LAB Task-02

 Implement Example of "malloc()" using "calloc()"

3. realloc():

"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 reallocate memory.

```
ptr = realloc(ptr, newSize);
```

Example: Memory Reallocation (realloc)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int i, * ptr, sum = 0;
ptr = malloc(100);
if (ptr == NULL) {
 printf("Error! memory not allocated.");
exit(0);
}

ptr = realloc(ptr,500);
if(ptr != NULL) printf("Memory created successfully\n");
return 0;
}
```

4. free()

"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 their own. Hence the free() method is used, whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.

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

int main() {
 int *ptr;
 int n = 5; // Number of elements

 // Allocate memory using malloc
 ptr = (int *)malloc(n * sizeof(int));

 // Check if memory allocation was successful
 if (ptr == NULL) {
 printf("Memory allocation failed.\n");
 return 1;
 }

 // Assign values to the array
 for (int i = 0; i < n; i++) {
 ptr[i] = (i + 1) * 10; // Storing 10, 20, 30, ...
 }

 // Display the values
 printf("Stored values:\n");
 for (int i = 0; i < n; i++) {
 printf("%d ", ptr[i]);
 }
 printf("\n");

 // Free the allocated memory
 free(ptr);
 printf("Memory has been successfully freed.\n");

 return 0;
     }
```

LAB Task-03

Implement free() function to malloc() program.