**Exercise 2: Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc.**

**C malloc() method**

**Program:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

// This pointer will hold the

// base address of the block created

int\* ptr;

int n, i;

// Get the number of elements for the array

printf("Enter number of elements:");

scanf("%d",&n);

printf("Entered number of elements: %d\n", n);

// Dynamically allocate memory using malloc()

**ptr = (int\*)malloc(n \* sizeof(int));**

// Check if the memory has been successfully

// allocated by malloc or not

if (ptr == NULL) {

printf("Memory not allocated.\n");

exit(0);

}

else {

// Memory has been successfully allocated

printf("Memory successfully allocated using malloc.\n");

// Get the elements of the array

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

ptr[i] = i + 1;

}

// Print the elements of the array

printf("The elements of the array are: ");

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

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

}

}

return 0;

}

**Calloc() method**

**Program**

#include <stdio.h>

#include <stdlib.h>

int main()

{

// This pointer will hold the

// base address of the block created

int\* ptr;

int n, i;

// Get the number of elements for the array

n = 5;

printf("Enter number of elements: %d\n", n);

// Dynamically allocate memory using calloc()

**ptr = (int\*)calloc(n, sizeof(int));**

// Check if the memory has been successfully

// allocated by calloc or not

if (ptr == NULL) {

printf("Memory not allocated.\n");

exit(0);

}

else {

// Memory has been successfully allocated

printf("Memory successfully allocated using calloc.\n");

// Get the elements of the array

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

ptr[i] = i + 1;

}

// Print the elements of the array

printf("The elements of the array are: ");

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

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

}

}

return 0;

}

**C free() method**

**Program:**

#include <stdio.h>

#include <stdlib.h>

int main() {

int n, i, \*ptr, sum = 0;

printf("Enter number of elements: ");

scanf("%d", &n);

**ptr = (int\*) malloc(n \* sizeof(int));**

// if memory cannot be allocated

if(ptr == NULL) {

printf("Error! memory not allocated.");

exit(0);

}

printf("Enter elements: ");

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

scanf("%d", ptr + i);

sum += \*(ptr + i);

}

printf("Sum = %d\n", sum);

// deallocating the memory

free(ptr);

printf("After free %d", \*ptr);

return 0;

}

**C realloc() method**

**Program:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

// This pointer will hold the

// base address of the block created

int\* ptr;

int n, i;

// Get the number of elements for the array

n = 5;

printf("Enter number of elements: %d\n", n);

// Dynamically allocate memory using calloc()

**ptr = (int\*)calloc(n, sizeof(int));**

// Check if the memory has been successfully

// allocated by malloc or not

if (ptr == NULL) {

printf("Memory not allocated.\n");

exit(0);

}

else {

// Memory has been successfully allocated

printf("Memory successfully allocated using calloc.\n");

// Get the elements of the array

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

ptr[i] = i + 1;

}

// Print the elements of the array

printf("The elements of the array are: ");

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

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

}

// Get the new size for the array

n = 10;

printf("\n\nEnter the new size of the array: %d\n", n);

// Dynamically re-allocate memory using realloc()

**ptr = realloc(ptr, n \* sizeof(int));**

// Memory has been successfully allocated

printf("Memory successfully re-allocated using realloc.\n");

// Get the new elements of the array

for (i = 5; i < n; ++i) {

ptr[i] = i + 1;

}

// Print the elements of the array

printf("The elements of the array are: ");

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

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

}

free(ptr);

}

return 0;

}