**EXPT. NO: 11**

**AIM:**

Program to demonstrate the use of malloc(), calloc(), realloc() and free() functions.

**Solution:**

**Program**

//C Program to demonstrate the use of realloc() function

#include <stdio.h>

#include <stdlib.h>

int main() {

    int\* ptr;   // This pointer will hold the base address of the block created

    int n, i;

    n = 5;   // Get the number of elements for the array

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

    ptr = (int\*)calloc(n, sizeof(int));   // Dynamically allocate memory using calloc()

    // Check if the memory has been successfully allocated by calloc or not

    if (ptr == NULL) {

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

        exit(0);

    }

    else {

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

        // 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]);

        }

        n = 10;    // Get the new size for the array

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

        ptr = realloc(ptr, n \* sizeof(int));   // Dynamically re-allocate memory using realloc()

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

        // 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;

}