Batch: **D2** Roll No.: **16010221038**

Experiment / assignment / tutorial No. **9**

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

**TITLE:**  Dynamic Memory Allocation.

**AIM:** Program to demonstrate dynamic memory allocation using malloc() & free () function.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Expected OUTCOME of Experiment:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.
4. [**http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**](http://cse.iitkgp.ac.in/~rkumar/pds-vlab/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

Create memory for int, char and float variables at run time using malloc() function and before exiting the program release the memory allocated at run time by using free() function.

**Algorithm:**

Step 1: Start.

Step 2: Declare an integer pointer.

Step3: Declare a char pointer.

Step4: Declare a float pointer.

Step5: Allocating memory to each pointer using dynamic memory allocation.

Step6: Terminating if value is Null.

Step7: Storing the entered value by user in pointers.

Step8: Using free function to erase the memory in space.

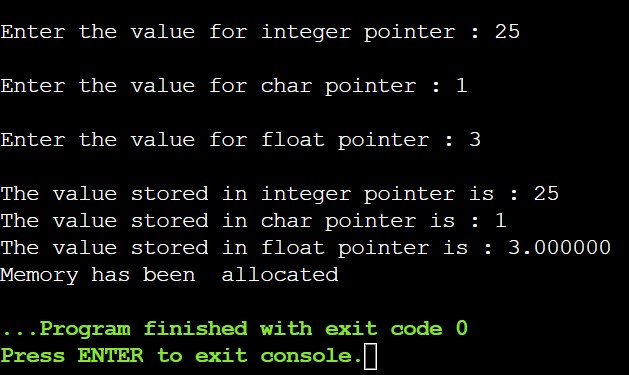
**Implementation details:**



**Output(s):**

**C**

**onclusion:**



**Hence, we have learned the use of malloc and calloc functions.**

**Post Lab Descriptive Questions**

1. **What is the difference between malloc and calloc?**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **malloc()** | **calloc()** |
|  |  |  |
| 1. | malloc() function creates a single block of memory of a specific size. | calloc() function assigns multiple blocks of memory to a single variable. |
|  |  |  |
| 2. | The number of arguments in malloc() is 1. | The number of arguments in calloc() is 2. |
| 3. | malloc() is faster. | calloc() is slower. |
|  |  |  |
| 4. | malloc() has high time efficiency. | calloc() has low time efficiency. |
|  |  |  |
| 5. | The memory block allocated by malloc() has a garbage value. | The memory block allocated by calloc() is initialized by zero. |
| 6. | malloc() indicates memory allocation. | calloc() indicates contiguous allocation |

1. **Consider the following C code. What will be the output?**

# include<stdio.h> # include<stdlib.h> void fun(int \*a)

{

a = (int\*)malloc(sizeof(int));

} int main() { int \*p; fun(p); \*p = 6; printf("%d\n",\*p); return(0);

}

* 1. Compiler Error
  2. 6
  3. Runtime Error (D) Garbage Value

**ANSWER: (C) Runtime Error.**

1. **Difference between Static and Dynamic Memory allocation**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Static Memory Allocation** | **Dynamic Memory Allocat** | **ion** |
| Static Memory Allocation memory is allocated at compile time. | Dynamic Memory Allocation memory is allocated at run tim | e. |
| Memory can not be Changed while executing a program. | memory can be Changed while executing a program. |  |
| Used in an array. | Used in the linked list. |  |
| It is fast and saves running time. | It is a bit slow. |  |
| It allocates memory from the stack. | It allocates memory from the h | eap |
| Allocated memory stays from start to end of the program. | Memory can be allocated at an time and can be released at any time. | y |
| It is less efficient than the  Dynamic allocation strategy. | It is more efficient than the Sta allocation strategy. | tic |
| Implementation of this type of allocation is simple. | Implementation of this type of allocation is complicated. |  |

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**