**Batch: A2 Roll No.:16010121045**

**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:**

CO 4: Design modular programs using functions and demonstrate the concept of pointers and file handling.

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

**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:**

Implementing a C program to create a student list of a class using Dynamic memory allocation. It will have the details of students as roll number and name. Program should support the following operations (menu driven).

1. Insert

2. Delete

3. Display

use malloc for insert and free for delete

**Algorithm:**

Step 1: Declare n=0,i,j in global scope.
Step 2: In main, declare pointer to an integer array and a character array.

Step 3: Ask for their choice of operation(ch).
Step 4: Take the choice and using switch case call the respective function.

* **Inserting**
1. We give two parameters to the function( pointer to array a and b)
2. We take the value from the user for number of insertions to take place
3. Then by using the for loop we insert the number of elements
4. In the end we call the printing function to see the output
* **Deleting**
1. We give two parameters to the function (pointer to array a and b)
2. Ask user for the name of person to be deleted
3. Search array arr and if the name matches, we use the index to free the elements to delta the elements in both the array with that index
4. After we rearrange the number to again form a continuous array
5. In the end we call a printing function to see the output.
* **Printing**
1. We give two parameters to the function (pointer to array a and b)
2. We run a for loop to iterate through both the arrays and print the element with that index.

**Implementation details:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int n=0;

void delete(int \*a[],char \*b[])

{

 char name[40];

 printf("\nEnter the name of the player you want to delete :- ");

 scanf("%s",name);

 int pos=0;

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

 {

 if(strcmp(b[i],name)==0)

 {

 pos=i;

 free(a[pos]);

 free(b[pos]);

 break;

 }

 }

 n--;

 for(int j=pos;j<n;j++)

 {

 a[j]=a[j+1];

 b[j]=b[j+1];

 }

}

void insert(int \*a[],char \*b[])

{

 int n1;

 printf("\nEnter Number of Students: ");

 scanf("%d",&n1);

 printf("\nEnter Roll number and Name :- \n");

 printf("eg: 143 Pargat\n");

 for(int j=0;j<n1;j++,n++)

 {

 a[n]=(int \*)malloc(sizeof(int));

 b[n]=(char \*)malloc(sizeof(char));

 scanf("%d %s",a[n],b[n]);

 }

}

void print(int \*a[],char \*b[])

{

 printf("\n\nName\tRoll-No\n");

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

 printf("%d\t%s\n",\*a[i],b[i]);

 printf("\n\n");

}

int main()

{

 int \*a[10];

 char \*b[10];

 int x=0;

 while(x!=4)

 {

 printf("\n\nList :\n");

 printf("1. Inserting\n");

 printf("2. Deleting\n");

 printf("3. Printing\n");

 printf("4. Exit\n");

 scanf("%d",&x);

 switch(x)

 {

 case 1: insert(a,b);

 break;

 case 2: delete(a,b);

 break;

 case 3: print(a,b);

 break;

 case 4: printf("Exit\n");

 break;

 default: printf("Please Enter a correct Option\n");

 break;

 }

 }

}

**Output(s):**





**Conclusion:**

We learned what a pointer is and how to use, learned how to use dynamic memory allocation and how to use malloc (), free(), calloc() functions.

**Post Lab Descriptive Questions**

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

|  |  |
| --- | --- |
| malloc() | calloc() |
| malloc() function creates a single block of memory of a specific size. | calloc() function assigns multiple blocks of memory to a single variable. |
| The number of arguments in malloc() is 1. | The number of arguments in calloc() is 2. |
| malloc() is faster. | calloc() is slower. |
| malloc() has high time efficiency. | calloc() has low time efficiency. |
| The memory block allocated by malloc() has a garbage value. | The memory block allocated by calloc() is initialized by zero. |
| 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);

}

(A) Compiler Error

(B) 6

(C) Runtime Error

(D) Garbage Value

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

|  |  |
| --- | --- |
| Static Memory Allocation | Dynamic Memory Allocation |
| In the static memory allocation, variables get allocated permanently, till the program executes or function call finishes. | In the Dynamic memory allocation, variables get allocated only if your program unit gets active. |
| Static Memory Allocation is done before program execution. | Dynamic Memory Allocation is done during program execution |
| It uses a [stack](https://www.geeksforgeeks.org/stack-data-structure/) for managing the static allocation of memory | It uses a [heap](https://www.geeksforgeeks.org/heap-data-structure/) for managing the dynamic allocation of memory |
| It is less efficient | It is more efficient |
| In Static Memory Allocation, there is no memory re-usability | In Dynamic Memory Allocation, there is memory re-usability and memory can be freed when not required |

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