K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent College of Somaiya Vidyavihar University)

Batch: G3 Roll No.:16010421063
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/**

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 respectiving function

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

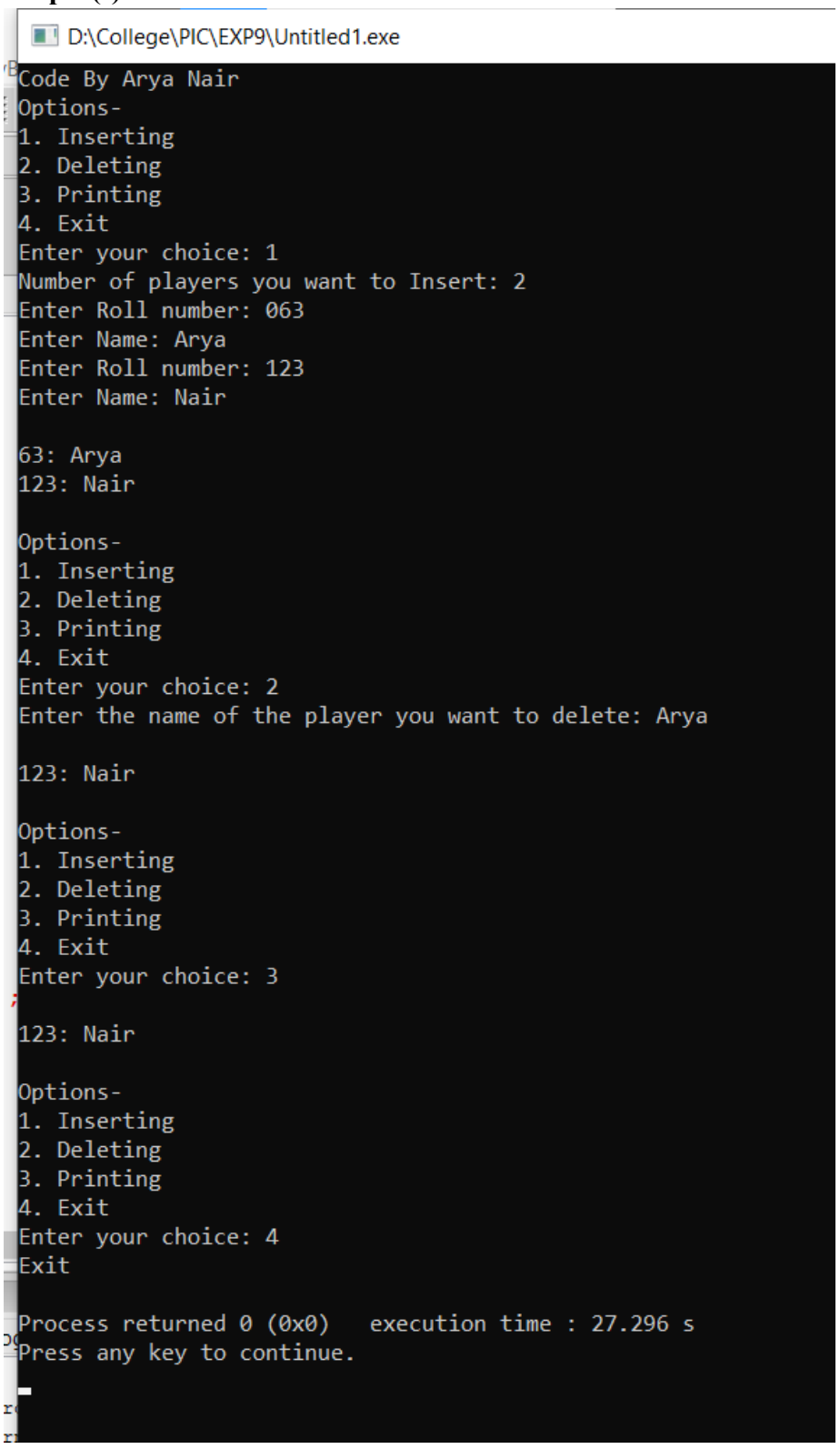
Implementation details:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int n=0,j,i;

void printing(int *a[],char *b[])
{
    printf("\n");
    for( i=0;i<n;i++)
    {
        printf("%d: %s\n",*a[i],b[i]);
    }
    printf("\n");
}

void inserting(int *a[],char *b[])
{
    int temp=0;
    printf("Number of players you want to Insert: ");
    scanf("%d",&temp);

    for( j=0;j<temp;j++,n++)
    {

        a[n]=(int *)malloc(sizeof(int));
        b[n]=(char *)malloc(sizeof(char));
        printf("Enter Roll number: ");
        scanf("%d",a[n]);
        printf("Enter Name: ");
        scanf("%s",b[n]);
    }
    printing(a,b);

}
void deleting(int *a[],char *b[])
{
    char name[40];
    printf("Enter the name of the player you want to delete:
```

```
");
    scanf("%s",name);
    int pos=0;
    for( i=0;i<n;i++)
    {
        if(strcmp(b[i],name)==0)
        {
            pos=i;
            free(a[pos]);
            free(b[pos]);
            break;
        }
    }
    n--;
    for( j=pos;j<n;j++)
    {
        a[j]=a[j+1];
        b[j]=b[j+1];
    }
    printing(a,b);
}


int main()
{
    int x=0;

    int *a[50];
    char *b[50];
    printf("Code By Arya Nair\n");
    while(x!=4)
    {
        printf("Options-\n");
        printf("1. Inserting\n");
        printf("2. Deleting\n");
        printf("3. Printing\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&x);
```

```
        switch(x)
        {
            case 1: inserting(a,b);
                    break;
            case 2: deleting(a,b);
                    break;
            case 3: printing(a,b);
                    break;
            case 4: printf("Exit\n");
                    break;
            default: printf("Incorrect input\n");
                     x=4;

        }
    }

}
```

Output(s):


D:\College\PIC\EXP9\Untitled1.exe
Code By Arya Nair
Options-
1. Inserting
2. Deleting
3. Printing
4. Exit
Enter your choice: 1
Number of players you want to Insert: 2
Enter Roll number: 063
Enter Name: Arya
Enter Roll number: 123
Enter Name: Nair
63: Arya
123: Nair
Options-
1. Inserting
2. Deleting
3. Printing
4. Exit
Enter your choice: 2
Enter the name of the player you want to delete: Arya
123: Nair
Options-
1. Inserting
2. Deleting
3. Printing
4. Exit
Enter your choice: 3
123: Nair
Options-
1. Inserting
2. Deleting
3. Printing
4. Exit
Enter your choice: 4
Exit
Process returned 0 (0x0) execution time : 27.296 s
Press any key to continue.

Conclusion:

We learned what a pointer is and how to use. We learned how to use dynamic memory allocation and how to use malloc () function. We learned what is the free () function and how to use it.

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.

2. **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

Ans- (C)Runtime Error

3. **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 for managing the static allocation of memory	It uses a heap 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
In static memory allocation, once the memory is allocated, the memory size can not change.	In dynamic memory allocation, when memory is allocated the memory size can be changed.

In this memory allocation scheme, we cannot reuse the unused memory.	This allows reusing the memory. The user can allocate more memory when required. Also, the user can release the memory when the user needs it.
In this memory allocation scheme, execution is faster than dynamic memory allocation.	In this memory allocation scheme, execution is slower than static memory allocation
In this memory is allocated at compile time.	In this memory is allocated at run time.
In this allocated memory remains from start to end of the program.	In this allocated memory can be released at any time during the program.
Example: This static memory allocation is generally used for array.	**Example:** This dynamic memory allocation is generally used for linked list.

Date: _____ **Signature of faculty in-charge**