Experiment No. : 1

Title: Demonstrate the use of arrays, array of structure and pointers using C.

Batch: B2(SY_IT) **Roll No.: 16010423076** **Experiment No.: 1**

Aim: Implement and demonstrate the use of arrays, array of structure and pointers using C.

Resources needed: Turbo C/C++ editor and C compiler (Online/Offline)

Theory

1) Arrays

- An array is a collection of items stored at contiguous memory locations.
- All elements in an array are of the same data type.
- Arrays allow for easy access and manipulation of data.

Examples

```
// An array of integers
int numbers[5] = {1, 2, 3, 4, 5};

// Accessing elements
int firstNumber = numbers[0];
int secondNumber = numbers[1];
```


2) Structures

- A structure is a user-defined data type in C that allows you to combine data items of different kinds.
- Structures are used to represent a record.

Examples

```
// Defining a structure
struct Student {
    char name[50];
    int age;
    float marks;
};

// Creating a structure variable
struct Student student1 = {"John Doe", 20, 85.5};

// Accessing structure members
printf("Name: %s\n", student1.name); // John Doe
printf("Age: %d\n", student1.age); // 20
printf("Marks: %.2f\n", student1.marks); // 85.5
```

3) Array of Structure

You can create an array of structures to store multiple records.

Examples

```
// Defining a structure
struct Student {
    char name[50];
    int age;
    float marks;
};

// Creating an array of structures
struct Student students[3] = {
    {"Ritesh", 19, 88.0},
    {"Omkar", 21, 90.5},
    {"Dev", 20, 85.0}
};

// Accessing elements in the array of structures
printf("Name of first student: %s\n", students[0].name); // Alice
printf("Age of second student: %d\n", students[1].age); // 21
printf("Marks of third student: %.2f\n", students[2].marks); // 85.0
```


4) Pointers and Pointers to Structures

- A pointer is a variable that stores the memory address of another variable.
- Pointers to structures allow you to manipulate structure data more efficiently.

Examples

```
// Defining a structure
struct Student {
    char name[50];
    int age;
    float marks;
};

// Creating a structure variable
struct Student student1 = {"Ritesh Jha", 20, 85.5};

// Creating a pointer to the structure
struct Student *ptr = &student1;

// Accessing structure members using the pointer
printf("Name: %s\n", ptr->name); // John Doe
printf("Age: %d\n", ptr->age); // 20
printf("Marks: %.2f\n", ptr->marks); // 85.5
```

5) Functions and Function signature

- A function is a block of code that performs a specific task.
- The function signature includes the function name, return type, and parameters.

Examples

```
// Function to add two numbers
int add(int a, int b) {
    return a + b;
}

// Using the function
int result = add(5, 3); // 8
printf("Result: %d\n", result);
```

Activity : Implementing a C program to create a roll call list of a class **using array of structure concept**. It has the details of students as roll number and name. Program should support following operations.

1. **Insert into last position.**
2. **Delete from last position.**
3. **Search specific student.**
4. **Display complete list of student with details.**

Results: A C program depicting the correct behaviour of mentioned concept and capable of handling all possible exceptional conditions/inputs and the same is reflecting clearly in the output.

Program and Output :

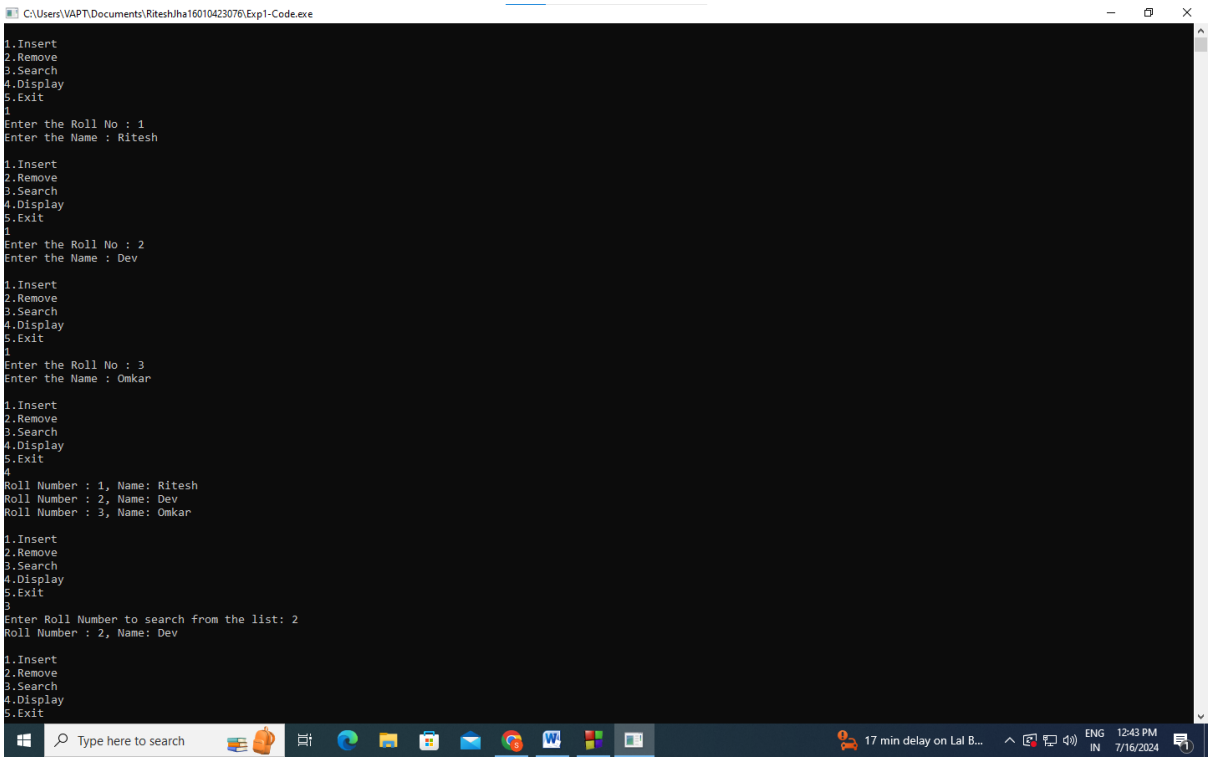
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int count = 0;
int i = 0;
int j = 0;
struct Student{
int rollNo;
char name[50];
}s[10];

void insertStudent(){
if (count<5){
printf("Enter the Roll No : ");
scanf("%d",&s[count].rollNo);
printf("Enter the Name : ");
scanf("%s",s[count].name);
count++;
}
else{
printf("List is full.\n");
}
}

void deleteStudent(){
if(count>0){
count--;
```

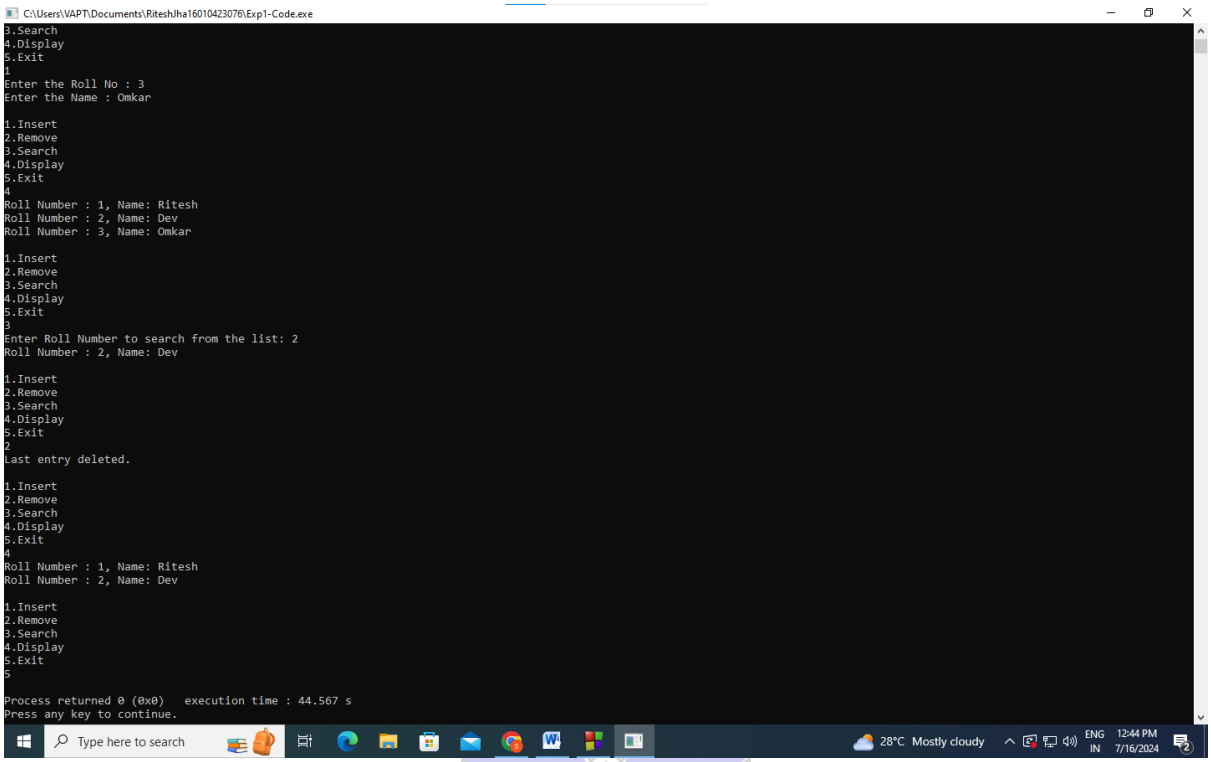
```
printf("Last entry deleted.\n");
}
else{
printf("List is empty.\n");
}
}

void searchStudent(){
int rollnumber;
printf("Enter Roll Number to search from the list: ");
scanf("%d",&rollnumber);

for(i=0;i<count;i++){
if(s[i].rollNo == rollnumber){
printf("Roll Number : %d, Name: %s \n", s[i].rollNo,s[i].name);
return;
}
}
printf("Student not present in the list\n");
}

void displayStudent()
{
if(count>0){
for(j=0;j<count;j++){
printf("Roll Number : %d, Name: %s\n", s[j].rollNo,s[j].name);
}
}
else{
```

```
printf("List empty\n");
}
}


int main(){
int k = 1;
while(k != 0){
 int n;
 printf("\n1.Insert\n2.Remove\n3.Search\n4.Display\n5.Exit\n");
 scanf("\n\n %d", &n);
 switch(n){
 case 1:
 insertStudent();
 break;
 case 2:
 deleteStudent();
 break;
 case 3:
 searchStudent();
 break;
 case 4:
 displayStudent();
 break;
 case 5:
 exit(0);
 break;
 default:
 printf("Error....\n");
```

```
 break;
 }
}
return 0;
}
```

Output :





Course Outcomes:

CO1. Comprehend the different data structures used in problem solving.

CO2. Apply linear and non-linear data structure in application development.

CO3. Implement concepts of advance data structures like set, map & dictionary..

CO4. Demonstrate sorting and searching methods.

Conclusion:

We started by reviewing the theory behind arrays, structures, and pointers, focusing on their syntax and functionality. We then applied this knowledge practically by writing functional code to perform various operations on data using structures.

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

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

- Y. Langsam, M. Augenstin and A. Tannenbaum, "**Data Structures using C**", Pearson Education Asia, 1st Edition, 2002

- **Data Structures A Psedocode Approach with C**, Richard F. Gilberg&Behrouz A. Forouzan, secondedition, CENGAGE Learning