Name	Lekh Sanatan Nayak
UID no.	2023800068
Experiment No.	4

AIM:	**Demonstrate the use of one-dimensional arrays to solve a given problem**
Program 1	
PROBLEM STATEMENT :	Write a C Program which contains a function to perform search of a particular element on an array. Create an array in main () and call the function to test it.
ALGORITHM:	1. Start 2. Declare variable ‘a’ and ‘size’. 3. Take user input to enter the size of array and read the size of user input. 4. Use a or loop to read ‘size’ integers from the user and store them in ‘a’. 5. Print the elements of the array 6. Take user input to search the number required. 7. Call the iselementpresent functio with array’a’, its size and the search number as its arguments 8. Inside the iselementpresent function, use a loop to iterate through the elements of the array 9. For each element in the array, check if it is equal to the search number. 10. If the number is found return 1 to indicate that the element is present. 11. If the number is not found return 0 to indicate that the element is absent. 12. In the main function, if the return value of iselementpresent is 1 print “Element found!” and is the return value is 0 print “Element not found. Better luck next time” 13. End

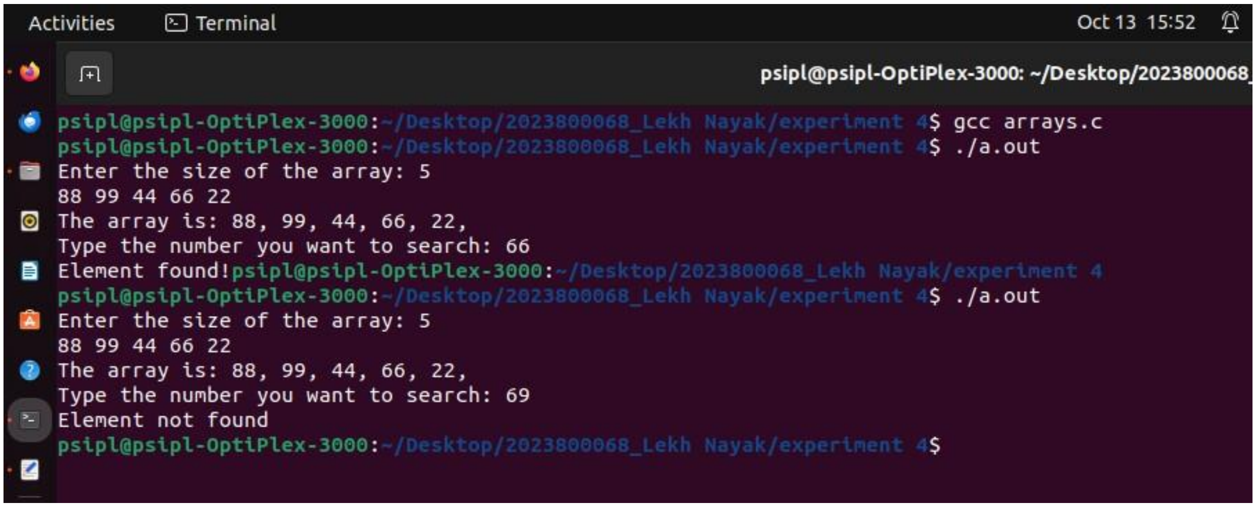
PROGRAM:

```
#include <stdio.h>
int iselementpresent(int a[],int size,int num) {
        for(int i=0;i<size;i++) {
                if (num==a[i]){
                        return 1;
                }
        }
return 0;
}

int main() {
        int size,search;
                printf("Enter the size of the array: ");
                scanf("%d",&size);
int a[size];
        for (int i=0;i<size;i++) {
                scanf("%d",&a[i]);
        }
        printf("The array is: ");
        for (int i=0;i<size;i++) {
                printf("%d, ",a[i]);
        }
        printf("\n");

        printf("Type the number you want to search: ");
        scanf("%d",&search);
        if(iselementpresent(a,size,search)==1){
                printf("Element found!");
        }
        else {
        printf("Element not found\nBetter luck next time");
        }
return 0;
}
```

RESULT:



```
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$ gcc arrays.c
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$ ./a.out
Enter the size of the array: 5
88 99 44 66 22
The array is: 88, 99, 44, 66, 22,
Type the number you want to search: 66
Element found!psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$ ./a.out
Enter the size of the array: 5
88 99 44 66 22
The array is: 88, 99, 44, 66, 22,
Type the number you want to search: 69
Element not found
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$
```

Program 2

PROBLEM STATEMENT :	Write a C Program which contains a function to sort array using selection sort. Create an array in main() and call the function to test it.
ALGORITHM:	1. Define a function “display” and use it to print the elements of an array and define another function “swap” used to swap two integer value. 2. Define “selectionsort” function which will perform the selectionsort algorithm to sort the array of elements. 3. Define “main” function and prompt the user to enter the size of the array and read the value into the ‘size’ variable. 4. Take user input to enter the elements of the array and read them into the array ‘a’ using a loop 5. Call the displaay function to print the unsorted array. 6. Call the selectionsort function to sort array ’a’ using the selection sort algorithm. 7. In function “selectionsort” initialize a variable min_index and set it to the first element 8. Start an outer loop that iterates through the array from the first element to the n-1. 9. Inside the outer loop create a inner loop to find the index of the minimum element in the unsorted part of the array (from i+1 to size-1) 10. Compare each element with the element at min_index, and if you find an element smaller than the current minimum, update min_index to the index of he smaller element. 11. After the inner loop if the element at a[i] is greater than the element

in a[min_index], swap the two elements using the “swap” function.

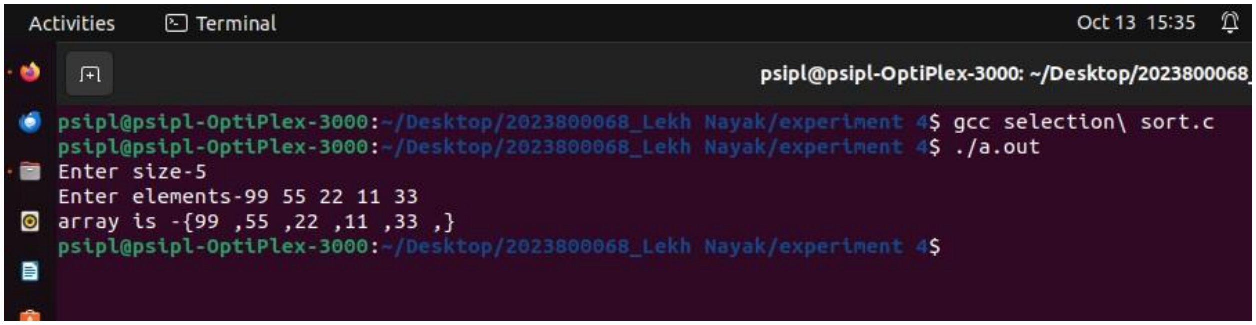
12. Continue this process untill the smallest unsorted element moves to its correct position.
13. After the outer loop completes the array is sorted in ascending order.
14. Print “SORTED” to indicate that the array is now sorted.
15. Call the “display” function again to print the sorted array.
16. end

PROGRAM:

```
#include<stdio.h>
void display(int a[], int size){
printf("array is -{");
    for (int i=0;i<size;i++){
        printf("%d ,",a[i]);
    }
    printf("}");
}
void swap(int *a,int *b){
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}
void selectionsort(int a[],int size){
    int min_index;
    for(int i=0;i<size;i++){
    min_index=i;
    for(int j=i+1;j<size;j++){
        if(a[j]<a[min_index])
            {
            min_index=j;
            }
    }
    if(a[i]>a[min_index]){
        swap(&a[i],&a[min_index]);
    }
}
}
int main(){
    int size;
    printf("Enter size-");
    scanf("%d",&size);
    int a[size];
    printf("Enter elements-");
    for(int i=0;i<size;i++){
        scanf("%d",&a[i]);
```

```
		}
		display(a,size);
		selectionsort(a,size);
		printf("\nSORTED ");
		display(a,size);
return 0;
}
```

RESULT:



```
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$ gcc selection\ sort.c
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$ ./a.out
Enter size-5
Enter elements-99 55 22 11 33
array is -{99 ,55 ,22 ,11 ,33 ,}
psipl@psipl-OptiPlex-3000:~/Desktop/2023800068_Lekh Nayak/experiment 4$
```

CONCLUSION: In this experiment we learnt about the use of one-dimensional arrays to solve a given problem