Program 1: Write a menu-driven Program to either insert, delete, search update or traverse an array.

Pseudocode:

Main Function

```
Display the following Menu:
Array operations:
1. Creation
2. Insertion
3. Deletion
4. Search and Update
5. Traversing
6. Exit”

Take size of array as input.
Declare an array of size 2*n.

While true
    {
    Take choice as input from the user.

    Switch (ch)
    {
    Call respective functions according to the choice from the user.

        If the choice is 2 or 3, ask another choice for doing the operation
        using ‘value’ or ‘position’ and call respective functions.

        Each Function passes the base address of the array and its size

    If the case is not present, show an ‘Invalid Choice’ message.
    }
    }
```

Creation Function

Take n elements of the array as input and store each element in the array via its address.

Search and Update Function

Take Search element and Updated Value as input from the user.

Search for the element in the array using a for loop and update it with another value if required.

Display a suitable message for the cases where search element is found or not.

Traversing Function

Using a for loop running from 0 to n display each element of the array using its address.

Insertion by Position Function

Take the value to be inserted and position where the value is to be placed as input.

Check if the position is not present in the array then return value of n along with a suitable message.

```
Run a for loop from n - 1 to position – 1:
{
                Right shift all the elements from position to n by one place.
}
```

Insert Given value at empty position and return value of n+1.

Insertion by Value Function

Take the Value to be Inserted and the Value after which to insert as input.

```
Run a for loop from 0 to 1:{
        if element is equal to inserted value
        {
                Run a for loop from n-1 to outer loop element
                {
                        Right shift all the elements from position to n by one
                    place.
                }
                Insert Given value at empty position of array.

                Increment the value of size of array by 1.
                }
            }
            Return value of size of array.
```

Deletion by Position Function

```
    Take the position of Value to be deleted as Input.

    Check if the position is not present in the array then return value of n along
    with a suitable message.

    Run a for loop from position of value to be deleted to n-1
    {
        Left shift all the elements from position to n-1 by one place.
    }
    Return value of size of array – 1.
```

Deletion by Value Function

```
    Take the Value to be deleted from the array as input.

    Run a for loop from 0 to n-1
    {
        if element is equal to inserted value
        {
            Run a for loop from n-1 to outer loop element
            {
                Left shift all the elements from position to n by one place.
            }
            Decrement the value of n by 1.
        }
    }
    Return value of size of array.
```

Source Code:

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <strings.h>

void creation(int *, int);
void search_and_update(int *, int);
void traversing(int *, int);
int insertion_by_pos(int *, int);
int insertion_by_val(int *, int);
int deletion_by_pos(int *, int);
int deletion_by_val(int *, int);

void main()                     // Main Function (Called at start of execution)
{
   int n, ch, ch1;
   printf("Array operations: \n1.Creation(Compulsory) \n2.Insertion \n3.Deletion \n
   4.Search & Update \n5.Traversing \n6.Exit \n\n");
   printf("Enter Size of array:");
   scanf("%d",&n);
   int arr[2*n];
   while(true)
   {
      printf("Enter Choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1: creation(arr,n);
               break;
         case 2: printf("1.Insertion By Position\n2.Insertion By Value\n");
               printf("Enter Choice:");
               scanf("%d",&ch1);
               switch(ch1)
               {
                  case 1: n = insertion_by_pos(arr,n);
                        break;
                  case 2: n = insertion_by_val(arr,n);
                        break;
                  default:printf("Invalid Choice\n");
```

```
                }
                break;
          case 3: printf("1.Deletion By Position\n2.Deletion By Value\n");
                printf("Enter Choice:");
                scanf("%d",&ch1);
                switch(ch1)
                {
                   case 1: n = deletion_by_pos(arr,n);
                         break;
                   case 2: n = deletion_by_val(arr,n);
                         break;
                   default:printf("Invalid Choice\n");
                }
                break;
          case 4: search_and_update(arr,n);
                break;
          case 5: traversing(arr,n);
                break;
          case 6: exit(0);

          default:printf("Invalid Choice\n");
       }
       printf("\nArray operations:\n1.Creation
(Compulsory)\n2.Insertion\n3.Deletion\n4.Search &
Update\n5.Traversing\n6.Exit\n\n");
    }
}

void creation(int *add, int n)          // Creation of Array (Compulsory)
{
    printf("Enter %d elements:\n",n);
    for(int i=0;i<n;i++)
    {
       scanf("%d",(add++));
    }
}

void search_and_update(int *add, int n) // Searching for a value and updating it
{
    int se, count=0, val;
    char str[5];
    printf("Enter Search Element:");
    scanf("%d",&se);
    printf("Do You Want to update the Searched Element(Yes or No):");
    scanf("%s",str);
    if(strcasecmp(str, "yes") == 0)
    {
       printf("Enter Updated Element Value:");
       scanf("%d",&val);
    }
    for(int i=0; i<n; i++)
    {
       if(add[i]==se)
          {
             count++;
             if(strcasecmp("yes", str) == 0)
                add[i] = val;
          }
    }
    if(count>0)
       printf("Search Element present %d times in array\n",count);
    else
       printf("Search Element not present in array\n");
}

void traversing(int *add, int n)        // Displaying the Array
{
    for(int i=0;i<n;i++)
    {
       printf("%d\t",add[i]);
    }
    printf("\n");
}

int insertion_by_pos(int *add, int n)   // Inserting an element at a position
{
    int val, pos, *add1, i;
    printf("Enter Value to be Inserted:");
    scanf("%d",&val);
    printf("Enter Position to Insert the Value at(Considering 1st element has 1st
    position):");
    scanf("%d",&pos);
    if((pos >= 0 && pos < n) == false)
       {
            printf("Invalid Position Entered\n");
            return n;
        }

    for(i = n-1; i >= pos-1; i--)
    {
        add[i+1]=add[i];
    }
    add[pos-1]=val;
    return n+1;
}

int insertion_by_val(int *add, int n)   // Inserting an element after another value
{
    int val, value, flag = 0;
    printf("Enter Value to be Inserted : ");
    scanf("%d",&value);
    printf("Enter Value after which to insert %d : ",value);
    scanf("%d",&val);
    for(int i = 0; i < n; i++)
    {
        if(add[i] == val)
        {
            flag = 1;
            for(int j = n-1; j > i; j--)
            {
                add[j+1] = add[j];
            }
            add[i+1] = value;
            n++;
        }
    }
    if(flag == 0)
        printf("Value not present in Array\n");
    return n;
}

int deletion_by_pos(int *add, int n)    // Deleting an element from a position
{
    int pos;
    printf("Enter Position to Delete the Value from(Considering 1st element has 1st
    position):");
```

```
   scanf("%d",&pos);
   if((pos >= 0 && pos < n) == false)
      {
         printf("Invalid Position Entered\n");
         return n;
      }
   for(int i = pos; i < n; i++)
   {
      add[i-1] = add[i];
   }
   return --n;
}

int deletion_by_val(int *add, int n)    // Deleting certain values from the array
{
   int val, flag = 0;
   printf("Enter Values to be deleted from the Array:");
   scanf("%d",&val);
   for(int i = 0; i < n; i++)
   {
      if(add[i] == val)
      {
         flag = 1;
         for(int j=i; j<n; j++)
         {
            add[j] = add[j+1];
         }
         n--;
      }
   }
   if(flag == 0)
      printf("Value not present in Array\n");
   return n;
}
```

Input/Output:

I. **Creation of Array:**

```
Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Size of array:5
Enter Choice:1
Enter 5 elements:
1 2 3 4 5

Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Choice:5
1	2	3	4	5
```

II. **Insertion of Array by position**

```
Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit
```

```
Enter Choice:2
1.Insertion By Position
2.Insertion By Value
Enter Choice:1
Enter Value to be Inserted:8
Enter Position to Insert the Value at(Considering 1st element has 1st
position):3

Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Choice:5
1       2       8       3       4       5
```

III. **Deletion of Array by Value**

```
Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Choice:3
1.Deletion By Position
2.Deletion By Value
Enter Choice:2
Enter Values to be deleted from the Array:3
```

```
Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Choice:5
1       2       8       4       5
```

IV. Search and Update

```
Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Choice:4
Enter Search Element:8
Do You Want to update the Searched Element(Yes or No):yes
Enter Updated Element Value:7
Search Element present 1 times in array

Array operations:
1.Creation (Compulsory)
2.Insertion
3.Deletion
4.Search & Update
5.Traversing
6.Exit

Enter Choice:5
1       2       7       4       5
```