Aim:

Implement an Array List Abstract Data Type (ADT) in C that supports the following operations:

- Insert an element at a specified position.
- Delete the first occurrence of a specified element.
- Display the elements of the list.
- Exit the program.

Input Format:
The program takes user input in a menu-driven way:
The program repeatedly prompts the user for a choice:

```
1. Insert
2. Delete
3. Display
4. Exit
```

The program prompts the user to "Enter choice: "

For Insert, the user will be prompted to enter element and position:

```
element to insert:
Enter the position:
```

For Delete, prompt for the element:

```
element to delete:
```

For Display, print the list as described.

Output Format:
Insert Operation:
If the specified position is invalid:

```
Invalid position
```

If the insertion is successful:

```
Inserted at position X
```

Delete Operation:
If the specified element is not found in the list:

```
Not found
```

If the deletion is successful:

```
Deleted successfully
```

Display Operation:

If the list is empty:

```
List is empty
```

If the list contains elements:

```
Elements in the list:
<elements separated by space>
```

Exit Operation:

- When the exit option is chosen, the program terminates without any additional output.

Note:

- Refer to the visible test cases for better understanding regarding Input and Output Formats.
- If the user enters a menu choice that is not recognized:

```
Invalid choice
```

Source Code:

array_list.c

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



struct ListADT {
int*array_list;
int size;
};

void initialize(struct ListADT *list)
{list->array_list = NULL;
list->size=0;
}
void insert_element(struct ListADT*list,int element,int position){
	if(position<0 ||position>list->size)
{
	printf("Invalid position\n");
}else{
	list->size++;
	list->array_list = (int*)realloc(list->array_list,list->size*sizeof(in
t));
for(int i = list->size-1;i>position;i--){list->array_list[i] = list->array_lis
t[i-1];


}
list->array_list[position] = element;
printf("Inserted at position %d\n",position);
}
}
```

```
void delete_element(struct ListADT*list,int element){
int found = 0;
for(int i = 0;i<list->size;i++){
if(list->array_list[i] == element){
found = 1;
for(int j = i;j<list->size - 1;j++){
list->array_list[j] = list->array_list[j + 1];
}
list->size--;
list->array_list=(int*)realloc(list->array_list,list->size*sizeof(int));
printf("Deleted successfully\n");
break;
}
}
if(!found){
printf("Not found\n");
}
}
void display(struct ListADT*list){
if(list->size == 0){
printf("List is empty\n");
}else{
printf("Elements in the list:\n");
for(int i = 0;i < list->size;i++) {
printf("%d ", list->array_list[i]);
}
printf("\n");
}
}
void free_list(struct ListADT *list){
free(list->array_list);
list->array_list = NULL;
list->size = 0;
}



int main() {
    struct ListADT my_list;
    initialize(&my_list);

    int choice;
    int element, position;

    while (1) {
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Exit\n");

        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("element to insert: ");
                scanf("%d", &element);
                printf("Enter the position: ");
                scanf("%d", &position);
                insert_element(&my_list, element, position);
                break;
            case 2:
                printf("element to delete: ");
                scanf("%d", &element);
                delete_element(&my_list, element);
                break;
            case 3:
                display(&my_list);
                break;
            case 4:
                free_list(&my_list);
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`1. Insert` 1
`2. Delete` 1
`3. Display` 1
`4. Exit` 1
`Enter choice:  1`
`element to insert:  1`
`Enter the position:  0`
`Inserted at position 0` 1
`1. Insert` 1
`2. Delete` 1
`3. Display` 1
`4. Exit` 1
`Enter choice:  1`
`element to insert:  2`
`Enter the position:  1`
`Inserted at position 1` 1
`1. Insert` 1
`2. Delete` 1
`3. Display` 1
`4. Exit` 1
`Enter choice:  1`
`element to insert:  4`
`Enter the position:  4`
`Invalid position` 3
`1. Insert` 3
`2. Delete` 3

```
3. Display 3
4. Exit 3
Enter choice:  3
Elements in the list: 2
1 2  2
1. Insert 2
2. Delete 2
3. Display 2
4. Exit 2
Enter choice:  2
element to delete:  3
Not found 2
1. Insert 2
2. Delete 2
3. Display 2
4. Exit 2
Enter choice:  2
element to delete:  1
Deleted successfully 2
1. Insert 2
2. Delete 2
3. Display 2
4. Exit 2
Enter choice:  2
element to delete:  2
Deleted successfully 3
1. Insert 3
2. Delete 3
3. Display 3
4. Exit 3
Enter choice:  3
List is empty 6
1. Insert 6
2. Delete 6
3. Display 6
4. Exit 6
Enter choice:  6
Invalid choice 4
1. Insert 4
2. Delete 4
3. Display 4
4. Exit 4
Enter choice:  4
```