DSA ASSIGNMENT

Assignment 1: Array operations

Name: Madhav Jha
Roll no.: 48
Branch: CSE (AI & ML)

Dynamic Array operations

Perform Following operations on Array use switch case

1. Create an array using dynamic memory allocation function
   store 999 at last position of an array
2. Insert an element in an array at specific position.
   shift elements at right side after inserting element in an array at specific position. and increase size of an array (using realloc function)
3. delete an element from array

   shift element to left side after deleting element
4. Find maximum number
5. Find minimum number
6. Linear search and binary search (don't traverse complete array traverse till you find 999 number)
7. Merging to array (Using Dynamic memory allocation)
8. Sorting

by call by reference

Code:

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

void printArray(int arr[], int n) {
    printf("[ ");
    for (int i = 0;i < n;i++) {
        printf("%d ", arr[i]);
    }
    printf("]\n");
}

void linearSearch(int arr[], int* n, int* num, int* index) {
    for (int i = 0;i < *n;i++) {
        if (arr[i] == *num) {
            *index = i;
            break;
        }
    }
}

int* bubbleSort(int arr[], int n) {
    for (int i = 0;i < n;i++) {
        for (int j = 0;j < n;j++) {
            if (arr[i] < arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    return arr;
}

void binarySearch(int arr[], int* n, int* num, int* index) {
    int l = 0, r = *n - 1, m;
    int count = 0;
    while (l <= r && count < 100) {
        m = (l + r) / 2;
        if (arr[m] == *num) {
```

```
            *index = m;
            break;
        }
        else if (arr[m] < *num) {
            l = m + 1;
        }
        else {
            r = m;
        }
        count++;
    }
}

void minMax(int arr[], int* n, int* max, int* min) {
    for (int i = 0;i < *n;i++) {
        if (arr[i] >= *max) {
            *max = arr[i];
        }
        if (arr[i] <= *min) {
            *min = arr[i];
        }
    }
}

void insertElem(int arr[], int* n, int* num, int* index) {
    *n += 1;
    arr = (int*)realloc(arr, (*n) * sizeof(int));
    for (int i = *n - 1;i > *index;i--) {
        arr[i] = arr[i - 1];
    }
    arr[*index] = *num;
}

int main() {
    int n = 2;
    printf("\nEnter number of elements: ");
    scanf("%d", &n);

    int* arr;
    arr = (int*)calloc(n, sizeof(int));
```

```
    printf("Enter array elements: ");
    for (int i = 0;i < n;i++) {
        scanf("%d", &arr[i]);
    }

    int en = 0;
    do {
        printf("\n\nChoose and option to perform: ");
        printf("\n1.Delete and element");
        printf("\n2.Linear search an element");
        printf("\n3.Sort the array");
        printf("\n4.Binary search an element");
        printf("\n5.Find minimum element");
        printf("\n6.Find maximum element");
        printf("\n7.Merge to array");
        printf("\n8.Insert an element");
        printf("\n0.Exit!!\nChoose option number: ");
        scanf("%d", &en);
        printf("\n");

        int num, index;
        int min = INT_MAX, max = -INT_MAX;
        int a1, a2, a3;
        int* arr1, * arr2, * arr3;
        switch (en)
        {
        case 0:
            printf("Exiting!!!");
            break;
        case 1:
            printf("Enter number to delete: ");
            scanf("%d", &num);

            index = -1;
            linearSearch(arr, &n, &num, &index);

            if (index == -1) {
                printf("Number not found!!!");
            }
            else {
```

```
            printf("Number found at index: %d", index);
            for (int i = index;i < n;i++) {
                arr[i] = arr[i + 1];
            }
            n--;
            printf("\nElement deleted: ");
            printArray(arr, n);
        }
        break;
    case 2:
        printf("Enter number to search: ");
        scanf("%d", &num);

        index = -1;
        linearSearch(arr, &n, &num, &index);

        if (index == -1) {
            printf("Number not found!!!");
        }
        else {
            printf("Number found at index: %d", index);
        }
        break;
    case 3:
        printf("Unsorted array: ");
        printArray(arr, n);

        bubbleSort(arr, n);
        printf("Sorted array: ");
        printArray(arr, n);
        break;
    case 4:
        printf("\nEnter number to search: ");
        scanf("%d", &num);

        index = -1;
        binarySearch(arr, &n, &num, &index);

        if (index == -1) {
            printf("Number not found!!!");
```

```
            }
            else {
                printf("Number found at index: %d", index);
            }
            break;
        case 5:
            min = INT_MAX, max = -INT_MAX;
            minMax(arr, &n, &max, &min);

            printf("\nMin element: %d", min);
            break;
        case 6:
            min = INT_MAX, max = -INT_MAX;
            minMax(arr, &n, &max, &min);

            printf("\nMax element: %d", max);
            break;
        case 7:
            a1 = 2;
            printf("\nEnter number of elements in array 1: ");
            scanf("%d", &a1);

            arr1 = (int*)calloc(a1, sizeof(int));
            printf("Enter array elements: ");
            for (int i = 0;i < a1;i++) {
                scanf("%d", &arr1[i]);
            }

            a2 = 2;
            printf("\nEnter number of elements in array 2: ");
            scanf("%d", &a2);

            arr2 = (int*)calloc(a2, sizeof(int));
            printf("Enter array elements: ");
            for (int i = 0;i < a2;i++) {
                scanf("%d", &arr2[i]);
            }

            a3 = a1 + a2;
            arr3 = (int*)calloc(a3, sizeof(int));
```

```
            for (int i = 0; i < a1; i++)
            {
                arr3[i] = arr1[i];
            }

            for (int i = 0; i < a2; i++)
            {
                arr3[a1 + i] = arr2[i];
            }
            printf("\nArray after merge: ");
            printArray(arr3, a3);
            break;
        case 8:
            printf("Enter number to insert: ");
            scanf("%d", &num);

            index = -1;
            printf("Enter index where to insert: ");
            scanf("%d", &index);

            if (index<0 || index>n) {
                printf("Index out of bound!!");
                break;
            }

            printf("\nArray before insertion: ");
            printArray(arr, n);

            insertElem(arr, &n, &num, &index);

            printf("\nArray After insertion: ");
            printArray(arr, n);
            break;
        default:
            break;
        }

    } while (en != 0);

    free(arr);
```

```
    return 0;
}
```

Output:

```
Enter number of elements: 6
Enter array elements: 3 1 5 4 7 6


Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 1

Enter number to delete: 4
Number found at index: 3
Element deleted: [ 3 1 5 7 6 ]


Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 2

Enter number to search: 1
Number found at index: 1
```

```
Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 3

Unsorted array: [ 3 1 5 7 6 ]
Sorted array: [ 1 3 5 6 7 ]


Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 4


Enter number to search: 5
Number found at index: 2
```

```
Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 5


Min element: 1

Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 6


Max element: 7
```

```
Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 7


Enter number of elements in array 1: 4
Enter array elements: 1 2 3 4

Enter number of elements in array 2: 4
Enter array elements: 4 5 6 7

Array after merge: [ 1 2 3 4 4 5 6 7 ]


Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 8
```

```
Enter number to insert: 8
Enter index where to insert: 4

Array before insertion: [ 1 3 5 6 7 ]

Array After insertion: [ 8000752 0 7995728 0 7 928333474 ]


Choose and option to perform:
1.Delete and element
2.Linear search an element
3.Sort the array
4.Binary search an element
5.Find minimum element
6.Find maximum element
7.Merge to array
8.Insert an element
0.Exit!!
Choose option number: 0

Exiting!!!
```