Aim ↬ To perform basic operations on 1-D arrays.

Objectives ↬

i. Write a program to create a 1-D array of unique Integers and perform the following
   a. Search a particular key value
   b. Delete a particular key value

ii. Write a program to create a 1-D array of Integers (containing repetition) and perform the following
   a. Search a particular key value
   b. Delete a particular key value

iii. Write a program to create a 1-D array of Integers and perform Bubble sort to arrange the elements in
   a. Ascending order
   b. Descending order

Software Required ↬ Visual Studio Code

Code 1 ↬

```
#include <stdio.h>
int main() {
   int arr[] = {5,4,3,7,6,1,8};
   int key,i,op,pos,flag = 0;
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("Original Array: ");
   for (i = 0; i<n;i++)
      printf("\n%d", arr[i]);
   printf("\nWhich operation is required? \n1)Search   2)Delete\n");
   scanf("%d", &op);
   if (op==1){
      printf("Enter the number to be searched: ");
      scanf("%d", &key);

      for (i=0; i<n; i++){
         if (arr[i]==key){
            flag = 1;
            break;
         }
      }
```

```
    if (flag == 1)
    printf("%d found at index: %d", key, i);
    else printf("%d not found.",key);
  }

  else if (op==2){
    printf("Enter the number to be deleted: ");
    scanf("%d", &key);

    for (i=0; i<n; i++){
      if (arr[i]==key){
        flag = 1;
        pos =i;
        break;
      }
    }
    if (flag == 1){
      for (i=pos; i<n-1; i++){
        arr[i] = arr[i+1];
      }
      printf("Array after deleting the element: ");
      for (i = 0; i<n-1;i++)
        printf("\n%d", arr[i]);
    }
    else
    printf("%d not found.",key);
  }

  else
  printf("Invalid input!");
  return 0;
}
```

Output ↬

```
Original Array:
5
4
3
7
6
1
8
Which operation is required?
1)Search   2)Delete
1
Enter the number to be searched: 7
7 found at index: 3
```

```
Original Array:
5
4
3
7
6
1
8
Which operation is required?
1)Search   2)Delete
1
Enter the number to be searched: 11
11 not found.
```

```
Original Array:
5
4
3
7
6
1
8
Which operation is required?
1)Search   2)Delete
2
Enter the number to be deleted: 3
Array after deleting the element:
5
4
7
6
1
8
```

Code 2 ↬

```
#include <stdio.h>
int main() {
   int arr[] = {5,4,5,7,6,4,8};
   int key,i,op,pos,flag,in = 0;
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("Original Array: ");
   for (i = 0; i<n;i++)
      printf("\n%d", arr[i]);
```

```
printf("\nWhich operation is required? \n1)Search   2)Delete\n");
scanf("%d", &op);
if (op==1){
   printf("Enter the number to be searched: ");
   scanf("%d", &key);
   for (i=0; i<n; i++){
      if (arr[i]==key){
         flag = 1;
         printf("%d found at index: %d\n", key, i);
      }
   }
   if (flag != 1)
   printf("%d not found!",key);
}

else if (op==2){printf("Enter the number to be deleted: ");
   scanf("%d", &key);
   printf("Enter the index of the number: ");
   scanf("%d", &in);

   for (i=0; i<n; i++){
      if (arr[i]==key & i==in){
         flag = 1;
         pos =i;
         break;
      }
   }

   if (flag == 1){
      for (i=pos; i<n-1; i++){
         arr[i] = arr[i+1];
      }
      printf("Array after deleting the element: ");
      for (i = 0; i<n-1;i++)
         printf("\n%d", arr[i]);
   }
   else
```

```
        printf("%d not found.",key);
    }

    else
    printf("Invalid input!");
    return 0;
}
```

Output ↬

```
Original Array:
5
4
5
7
6
4
8
Which operation is required?
1)Search   2)Delete
1
Enter the number to be searched: 4
4 found at index: 1
4 found at index: 5
```

```
Original Array:
5
4
5
7
6
4
8
Which operation is required?
1)Search   2)Delete
1
Enter the number to be searched: 2
2 not found!
```

```
Original Array:
5
4
5
7
6
4
8
Which operation is required?
1)Search   2)Delete
2
Enter the number to be deleted: 5
Enter the index of the number: 2
Array after deleting the element:
5
4
7
6
4
8
```

Code 3 ↬

```
#include <stdio.h>
int main() {
   int arr[] = {-2, 3 ,6, 8, -1, 7, -2};
   int i,j,temp = 0;
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("Original Array: ");
   for (i = 0; i<n; i++)
      printf("\n%d", arr[i]);

   for (i =0; i<n; i++){
      for(j = 0; j<(n-i-1); j++){
         if (arr[j] > arr[j+1]){
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }

   printf("\nArray in Ascending order: ");
   for (i = 0; i<n; i++)
      printf("\n%d", arr[i]);

   for (i =0; i<n; i++){
      for(j = 0; j<(n-i-1); j++){

        if (arr[j] < arr[j+1]){
                    temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
          }

   printf("\nArray in Descending order: ");
   for (i = 0; i<n; i++)
```

```
    printf("\n%d", arr[i]);
}
```

Output ↬

```
Original Array:
-2
3
6
8
-1
7
-2
Array in Ascending order:
-2
-2
-1
3
6
7
8
Array in Descending order:
8
7
6
3
-1
-2
-2
```

Result ↬

The programs were successfully implemented to perform basic operations on 1-D arrays, including searching for key values, deleting key values, and sorting arrays in ascending and descending order using the Bubble Sort algorithm. The outputs verified the correct execution of these array operations.

Conclusion ↬

The experiment successfully demonstrated key operations on 1-D arrays, including unique and repetitive element handling, key searching, deletion, and sorting using Bubble Sort. These operations are fundamental in array

manipulation and reinforce the understanding of algorithmic implementation in C programming.

Precautions ↬

- Ensure the array size and indices for deletion are within valid bounds to avoid accessing invalid memory locations.
- Carefully check array inputs for uniqueness when required and handle input edge cases like duplicates appropriately.
- For sorting, verify that comparisons and swaps are correctly implemented to achieve the desired order without data loss or errors.