NAME : KRIPANSH SRIVASTAVA

BATCH NO : 46

SAP ID: 500122597

ENROL.NO: R21422543

CREATION OF ARRAY

1) STATIC

```c
#include <stdio.h>

int main() {
    // Declare a static array Let the size of array be 5
    int arr[5] = {1, 2, 3, 4, 5};

    // Print the elements
    for(int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    fflush(stdout);/// to clear the buffer

    return 0;
}
```

FLUSH:- This function is used to clear output buffer.

OUTPUT :-

```
[Running] cd "c:\Users\Kripansh Srivastava\OneDrive - UPES\Desktop\DSA assignment\" && gcc DSAcodes.c -o DSAcodes &&
"c:\Users\Kripansh Srivastava\OneDrive - UPES\Desktop\DSA assignment\"DSAcodes
1 2 3 4 5
[Done] exited with code=0 in 0.489 seconds
```

- With Pointer

```c
#include <stdio.h>

int main() {
    // Declare a static array of size 5
    int arr[5] = {1, 2, 3, 4, 5};

    // Declare a pointer and point it to the start of the array
    int *ptr = arr;

    // Print the elements
    for(int i = 0; i < 5; i++) {
        printf("%d ", *(ptr + i));
    }
    printf("\n");  // Add a newline character to flush the buffer

    return 0;
}
```

- OUTPUT

```
[Running] cd "c:\Users\Kripansh Srivastava\OneDrive - UPES\Desktop\DSA assignment\" && gcc DSAcodes.c -o DSAcodes &&
"c:\Users\Kripansh Srivastava\OneDrive - UPES\Desktop\DSA assignment\"DSAcodes
1 2 3 4 5
```

- With Function and pointer

```c
#include <stdio.h>

// Function to print elements of an array
void printArray(int *arr, int size) {
    for(int i = 0; i < size; i++) {
        printf("%d ", *(arr + i));
    }
    printf("\n");  // Add a newline character to flush the buffer
}

int main() {
    // Declare a static array of size 5
    int arr[5] = {1, 2, 3, 4, 5};

    // Call the function to print the array
    printArray(arr, 5);

    return 0;
}
```

- OUTPUT

```
PROBLEMS    OUTPUT    DEBUG CONSOLE
[Running] cd "c:\Users\Kripans
"c:\Users\Kripansh Srivastava\
1 2 3 4 5
```

- With Function

DSAcodes.c > main()

```c
#include <stdio.h>

// Function to print elements of an array
void printArray(int arr[], int size) {
    for(int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");  // Add a newline character to flush the buffer
}

int main() {
    // Declare a static array of size 5
    int arr[5] = {1, 2, 3, 4, 5};

    // Call the function to print the array
    printArray(arr, 5);

    return 0;
}
```

- Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    SERIAL MO
[Running] cd "c:\Users\Kripansh Srivastava\OneDrive - UPE
"c:\Users\Kripansh Srivastava\OneDrive - UPES\Desktop\DSA
1 2 3 4 5
```

- Dynamic

dsa.c > main()

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

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array
        for(int i = 0; i < size; ++i) {
            array[i] = i + 1;
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d, ", array[i]);
        }
    }

    return 0;
}
```

- Output

```
Enter size of the array: 12
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,

...Program finished with exit code 0
Press ENTER to exit console.
```

- With Pointer

```c
dsa.c > main()
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array
        int *ptr = array;
        for(int i = 0; i < size; ++i, ++ptr) {
            *ptr = i + 1;
        }

        // Print the array elements
        printf("The elements of the array are: ");
        ptr = array;
        for(int i = 0; i < size; ++i, ++ptr) {
            printf("%d, ", *ptr);
        }
    }

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 8
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8,

...Program finished with exit code 0
Press ENTER to exit console.
```

- With Function

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

void fillAndPrintArray(int* array, int size) {
    // Fill the array
    for(int i = 0; i < size; ++i) {
        array[i] = i + 1;
    }

    // Print the array elements
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d, ", array[i]);
    }
}

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        fillAndPrintArray(array, size);
    }

    return 0;
}
```

- Output

```
Enter size of the array: 7
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7,
```

- With Pointer and Function

dsa.c > main()

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

void fillAndPrintArray(int* array, int size) {
    // Get elements of the array
    int *ptr = array;
    for(int i = 0; i < size; ++i, ++ptr) {
        *ptr = i + 1;
    }

    // Print the array elements
    printf("The elements of the array are: ");
    ptr = array;
    for(int i = 0; i < size; ++i, ++ptr) {
        printf("%d, ", *ptr);
    }
}

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        fillAndPrintArray(array, size);
    }

    return 0;
}
```

- Output

```
Enter size of the array: 8
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8,
```

TRAVERSING OF AN ARRAY

- WITHOUT POINTER AND FUNCTION

dsa.c > main()

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

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        printf("Enter elements of the array: ");
        for(int i = 0; i < size; ++i) {
            scanf("%d", &array[i]);
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d, ", array[i]);
        }
    }

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter elements of the array: 15 4 4 5
The elements of the array are: 15, 4, 4, 5,
```

- With pointer
-

dsa.c > main()

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

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        printf("Enter elements of the array: ");
        int *ptr = array;
        for(int i = 0; i < size; ++i, ++ptr) {
            scanf("%d", ptr);
        }

        // Print the array elements
        printf("The elements of the array are: ");
        ptr = array;
        for(int i = 0; i < size; ++i, ++ptr) {
            printf("%d, ", *ptr);
        }
    }

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter elements of the array: 5 6 8 9
The elements of the array are: 5, 6, 8, 9,

...Program finished with exit code 0
Press ENTER to exit console.
```

- With Function

dsa.c > main()

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

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        printf("Enter elements of the array: ");
        int *ptr = array;
        for(int i = 0; i < size; ++i, ++ptr) {
            scanf("%d", ptr);
        }

        // Print the array elements
        printf("The elements of the array are: ");
        ptr = array;
        for(int i = 0; i < size; ++i, ++ptr) {
            printf("%d, ", *ptr);
        }
    }

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter elements of the array: 8 7 5 6
The elements of the array are: 8, 7, 5, 6,

...Program finished with exit code 0
Press ENTER to exit console.
```

- With Pointers

dsa.c > main()

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

void getAndPrintArray(int* array, int size) {
    // Get elements of the array from the user
    printf("Enter elements of the array: ");
    int *ptr = array;
    for(int i = 0; i < size; ++i, ++ptr) {
        scanf("%d", ptr);
    }

    // Print the array elements
    printf("The elements of the array are: ");
    ptr = array;
    for(int i = 0; i < size; ++i, ++ptr) {
        printf("%d, ", *ptr);
    }
}

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        getAndPrintArray(array, size);
    }

    return 0;
}
```

And Functions

- Output

```
Enter size of the array: 7
Memory successfully allocated using malloc.
Enter elements of the array: 4 5 6 1 2 3 7
The elements of the array are: 4, 5, 6, 1, 2, 3, 7,

...Program finished with exit code 0
Press ENTER to exit console.
```

Insertion

1. In the Beginning

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

int main() {
    int *array;
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);

    // Allocate memory for the array
    array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        printf("Enter elements of the array: ");
        for(int i = 0; i < size; ++i) {
            scanf("%d", &array[i]);
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d, ", array[i]);
        }

        // Insert an element at the beginning of the array
        int *newArray = (int*) malloc((size + 1) * sizeof(int));
        if(newArray == NULL) {
            printf("Memory not allocated for the new array.\n");
            exit(0);
        }

        printf("\nEnter the new element to be inserted at the beginning: ");
        int newElement;
        scanf("%d", &newElement);

        newArray[0] = newElement;
        for(int i = 0; i < size; ++i) {
            newArray[i + 1] = array[i];
        }

        free(array);
        array = newArray;
        size++;

        // Print the new array elements
        printf("The elements of the new array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d, ", array[i]);
        }
    }

    return 0;
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter elements of the array: 5 8 9 6
The elements of the array are: 5, 8, 9, 6,
Enter the new element to be inserted at the beginning: 5
The elements of the new array are: 5, 5, 8, 9, 6,

...Program finished with exit code 0
Press ENTER to exit console.
```

- With Pointer

```c
int main() {
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i);
            scanf("%d", &array[i]);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Insert an element at the beginning of the array
        int *newArray = (int*) malloc((size + 1) * sizeof(int));
        if(newArray == NULL) {
            printf("Memory not allocated for the new array.\n");
            exit(0);
        }

        printf("\nEnter the new element to be inserted at the beginning: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        newArray[0] = newElement;
        for(int i = 0; i < size; ++i) {
            newArray[i + 1] = array[i];
        }

        free(array);
        array = newArray;
        size++;

        // Print the new array elements
        printf("The elements of the new array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 0: 7
Enter element 1: 8
Enter element 2: 9
Enter element 3: 2
The elements of the array are: 7, 8, 9, 2

Enter the new element to be inserted at the beginning: 7
The elements of the new array are: 7, 7, 8, 9, 2
```

- With Function

dsa.c > main()

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

void getArray(int array[], int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void insertAtBeginning(int array[], int size, int newElement) {
    for(int i = size; i > 0; --i) {
        array[i] = array[i - 1];
    }
    array[0] = newElement;
}

void printArray(int array[], int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", array[i]);
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at the beginning of the array
        printf("\nEnter the new element to be inserted at the beginning: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtBeginning(array, size, newElement);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 7
Enter element 2: 5
Enter element 3:  5
Enter element 4:  5
The elements of the array are: 7, 5, 5, 5

Enter the new element to be inserted at the beginning: 7
The elements of the array are: 7, 7, 5, 5, 5
```

- With Pointer And Function

dsa.c > main()

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

void getArray(int *array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", array + i);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void insertAtBeginning(int *array, int size, int newElement) {
    for(int i = size; i > 0; --i) {
        *(array + i) = *(array + i - 1);
    }
    *array = newElement;
}

void printArray(int *array, int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at the beginning of the array
        printf("\nEnter the new element to be inserted at the beginning: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtBeginning(array, size, newElement);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 6
Enter element 2: 63
Enter element 3: 36
Enter element 4: 69
Enter element 5: 96
The elements of the array are: 6, 63, 36, 69, 96

Enter the new element to be inserted at the beginning: 699
The elements of the array are: 699, 6, 63, 36, 69, 96
```

- IN the End

dsa.c > main()

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", &array[i]);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Insert an element at the end of the array
        printf("\nEnter the new element to be inserted at the end: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        array[size] = newElement;
        size++;

        // Print the new array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 8
Enter element 2: 8
Enter element 3: 6
Enter element 4: 1
Enter element 5: 5
The elements of the array are: 8, 8, 6, 1, 5

Enter the new element to be inserted at the end: 5
The elements of the array are: 8, 8, 6, 1, 5, 5
```

- With pointer

dsa.c > main()

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

void getArray(int *array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", array + i);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void insertAtBeginning(int *array, int size, int newElement) {
    for(int i = size; i > 0; --i) {
        *(array + i) = *(array + i - 1);
    }
    *array = newElement;
}

void printArray(int *array, int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at the beginning of the array
        printf("\nEnter the new element to be inserted at the beginning: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtBeginning(array, size, newElement);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 3
Memory successfully allocated using malloc.
Enter element 1: 5
Enter element 2: 69
Enter element 3: 96
The elements of the array are: 5, 69, 96

Enter the new element to be inserted at the end: 6
The elements of the array are: 5, 69, 96, 6
```

- With function

dsa.c

dsa.c > main()

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

void getArray(int array[], int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void insertAtEnd(int array[], int size, int newElement) {
    array[size] = newElement;
}

void printArray(int array[], int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", array[i]);
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at the end of the array
        printf("\nEnter the new element to be inserted at the end: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtEnd(array, size, newElement);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 5
Enter element 2: 2
Enter element 3: 5
Enter element 4: 32
The elements of the array are: 5, 2, 5, 32

Enter the new element to be inserted at the end: 2
The elements of the array are: 5, 2, 5, 32, 2
```

- With function and Pointer

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

void getArray(int* array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", array + i);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void insertAtEnd(int* array, int size, int newElement) {
    *(array + size) = newElement;
}

void printArray(int* array, int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at the end of the array
        printf("\nEnter the new element to be inserted at the end: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtEnd(array, size, newElement);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 2
Memory successfully allocated using malloc.
Enter element 1: 2
Enter element 2: 2
The elements of the array are: 2, 2

Enter the new element to be inserted at the end: 1
The elements of the array are: 2, 2, 1
```

- At Particular Location

```c
int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", &array[i]);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Insert an element at a particular location in the array
        printf("\nEnter the new element to be inserted: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        printf("Enter the location at which the new element should be inserted (0 to %d): ", size);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        // Shift the elements to the right of the insertion point
        for(int i = size; i > location; --i) {
            array[i] = array[i - 1];
        }

        // Insert the new element
        array[location] = newElement;
        size++;

        // Print the new array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 2
Enter element 2: 2
Enter element 3: 2
Enter element 4: 2
Enter element 5:
2
The elements of the array are: 2, 2, 2, 2, 2

Enter the new element to be inserted: 2
Enter the location at which the new element should be inserted (0 to 5): 5
The elements of the array are: 2, 2, 2, 2, 2, 2
```

- With Pointer

dsa.c > main()

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", array + i);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Insert an element at a particular location in the array
        printf("\nEnter the new element to be inserted: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        printf("Enter the location at which the new element should be inserted (0 to %d): ", size);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        // Shift the elements to the right of the insertion point
        for(int i = size; i > location; --i) {
            *(array + i) = *(array + i - 1);
        }

        // Insert the new element
        *(array + location) = newElement;
        size++;

        // Print the new array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);
```

- Output

```
Enter size of the array: 3
Memory successfully allocated using malloc.
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
The elements of the array are: 1, 2, 3

Enter the new element to be inserted: 2
Enter the location at which the new element should be inserted (0 to 3): 1
The elements of the array are: 1, 2, 2, 3
```

- With Function

```c
void getArray(int array[], int size) {
    }
}

void insertAtLocation(int array[], int size, int newElement, int location) {
    // Shift the elements to the right of the insertion point
    for(int i = size; i > location; --i) {
        array[i] = array[i - 1];
    }

    // Insert the new element
    array[location] = newElement;
}

void printArray(int array[], int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", array[i]);
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at a particular location in the array
        printf("\nEnter the new element to be inserted: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        printf("Enter the location at which the new element should be inserted (0 to %d): ", size);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtLocation(array, size, newElement, location);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 7
Enter element 2: 5
Enter element 3: 3
Enter element 4: 1
Enter element 5: 5
The elements of the array are: 7, 5, 3, 1, 5

Enter the new element to be inserted: 5
Enter the location at which the new element should be inserted (0 to 5): 1
The elements of the array are: 7, 5, 5, 3, 1, 5
```

- With Function and Pointer

dsa.c > main()

```c
void getArray(int *array, int size) {
    }
}

void insertAtLocation(int *array, int size, int newElement, int location) {
    // Shift the elements to the right of the insertion point
    for(int i = size; i > location; --i) {
        *(array + i) = *(array + i - 1);
    }

    // Insert the new element
    *(array + location) = newElement;
}

void printArray(int *array, int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc((size + 1) * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Insert an element at a particular location in the array
        printf("\nEnter the new element to be inserted: ");
        int newElement;
        scanf("%d", &newElement);
        // Flush the input buffer
        while (getchar() != '\n');

        printf("Enter the location at which the new element should be inserted (0 to %d): ", size);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        insertAtLocation(array, size, newElement, location);
        size++;

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 4
The elements of the array are: 1, 2, 3, 4

Enter the new element to be inserted:
1
Enter the location at which the new element should be inserted (0 to 4): 0
The elements of the array are: 1, 1, 2, 3, 4
```

Deletion Of an Element

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", &array[i]);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Delete an element at a particular location in the array
        printf("\nEnter the location of the element to be deleted (0 to %d): ", size - 1);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        // Shift the elements to the left of the deletion point
        for(int i = location; i < size - 1; ++i) {
            array[i] = array[i + 1];
        }
        size--;

        // Print the new array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 1
Enter element 2: 2
Enter element 3: 4
Enter element 4:
55
The elements of the array are: 1, 2, 4, 55

Enter the location of the element to be deleted (0 to 3): 2
The elements of the array are: 1, 2, 55
```

- WITH POINTER

dsa.c > main()

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", array + i);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Delete an element at a particular location in the array
        printf("\nEnter the location of the element to be deleted (0 to %d): ", size - 1);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        // Shift the elements to the left of the deletion point
        for(int i = location; i < size - 1; ++i) {
            *(array + i) = *(array + i + 1);
        }
        size--;

        // Print the new array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 6
Enter element 5: 5
The elements of the array are: 1, 2, 3, 6, 5

Enter the location of the element to be deleted (0 to 4): 3
The elements of the array are: 1, 2, 3, 5
```

- WITH FUNCTION

dsa.c > main()

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

void getArray(int array[], int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void deleteAtLocation(int array[], int *size, int location) {
    // Shift the elements to the left of the deletion point
    for(int i = location; i < *size - 1; ++i) {
        array[i] = array[i + 1];
    }
    (*size)--;
}

void printArray(int array[], int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", array[i]);
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Delete an element at a particular location in the array
        printf("\nEnter the location of the element to be deleted (0 to %d): ", size - 1);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        deleteAtLocation(array, &size, location);

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 2
Enter element 2: 4
Enter element 3: 5
Enter element 4: 1
The elements of the array are: 2, 4, 5, 1

Enter the location of the element to be deleted (0 to 3): 2
The elements of the array are: 2, 4, 1
```

- WITH POINTER AND FUNCTION

dsa.c > main()

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

void getArray(int *array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", array + i);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void deleteAtLocation(int *array, int *size, int location) {
    // Shift the elements to the left of the deletion point
    for(int i = location; i < *size - 1; ++i) {
        *(array + i) = *(array + i + 1);
    }
    (*size)--;
}

void printArray(int *array, int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Delete an element at a particular location in the array
        printf("\nEnter the location of the element to be deleted (0 to %d): ", size - 1);
        int location;
        scanf("%d", &location);
        // Flush the input buffer
        while (getchar() != '\n');

        deleteAtLocation(array, &size, location);

        // Print the new array elements
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 2
Enter element 2: 1
Enter element 3: 1
Enter element 4: 4
The elements of the array are: 2, 1, 1, 4

Enter the location of the element to be deleted (0 to 3): 2
The elements of the array are: 2, 1, 4
```

- Searching in an array

C dsa.c

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", &array[i]);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Search for an element in the array
        printf("\nEnter the element to be searched: ");
        int element;
        scanf("%d", &element);
        // Flush the input buffer
        while (getchar() != '\n');

        int found = 0;
        for(int i = 0; i < size; ++i) {
            if(array[i] == element) {
                printf("Element %d found at location %d\n", element, i);
                found = 1;
                break;
            }
        }

        if(!found) {
            printf("Element %d not found in the array\n", element);
        }
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

X

- OUTPUT

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 5
Enter element 2: 9
Enter element 3: 0
Enter element 4: 6
The elements of the array are: 5, 9, 0, 6

Enter the element to be searched: 9
Element 9 found at location 1
```

- WITH POINTERS

dsa.c > main()

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", array + i);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Search for an element in the array
        printf("\nEnter the element to be searched: ");
        int element;
        scanf("%d", &element);
        // Flush the input buffer
        while (getchar() != '\n');

        int found = 0;
        for(int i = 0; i < size; ++i) {
            if(*(array + i) == element) {
                printf("Element %d found at location %d\n", element, i);
                found = 1;
                break;
            }
        }

        if(!found) {
            printf("Element %d not found in the array\n", element);
        }
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- OUTPUT

```
Enter size of the array: 6
Memory successfully allocated using malloc.
Enter element 1: 2
Enter element 2: 5
Enter element 3: 8
Enter element 4: 9
Enter element 5: 6
Enter element 6: 3
The elements of the array are: 2, 5, 8, 9, 6, 3

Enter the element to be searched: 6
Element 6 found at location 4
```

- WITH FUNCTION

dsa.c > main()

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

void getArray(int *array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", array + i);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void printArray(int *array, int size) {
    printf("The elements of the array are: ");
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

int searchElement(int *array, int size, int element) {
    for(int i = 0; i < size; ++i) {
        if(*(array + i) == element) {
            return i;
        }
    }
    return -1;
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printArray(array, size);

        // Search for an element in the array
        printf("\nEnter the element to be searched: ");
        int element;
        scanf("%d", &element);
        // Flush the input buffer
        while (getchar() != '\n');

        int location = searchElement(array, size, element);

        if(location != -1) {
            printf("Element %d found at location %d\n", element, location);
        }
        else {
            printf("Element %d not found in the array\n", element);
        }
    }

    // Free the allocated memory
    free(array);
```

- OUTPUT

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 1
Enter element 2: 5
Enter element 3: 2
Enter element 4: 6
Enter element 5: 3
The elements of the array are: 1, 5, 2, 6, 3

Enter the element to be searched:
7
Element 7 not found in the array
```

- Sort the element of an array

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", &array[i]);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Sort the array using Bubble Sort
        for(int i = 0; i < size - 1; ++i) {
            for(int j = 0; j < size - i - 1; ++j) {
                if(array[j] > array[j + 1]) {
                    int temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }

        // Print the sorted array
        printf("The sorted elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", array[i]);
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 2
Enter element 2: 6
Enter element 3: 3
Enter element 4: 2
The elements of the array are: 2, 6, 3, 2
The sorted elements of the array are: 2, 2, 3, 6
```

- With pointer

dsa.c > main()

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

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        for(int i = 0; i < size; ++i) {
            printf("Enter element %d: ", i + 1);
            scanf("%d", array + i);
            // Flush the input buffer
            while (getchar() != '\n');
        }

        // Print the array elements
        printf("The elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");

        // Sort the array using Bubble Sort
        for(int i = 0; i < size - 1; ++i) {
            for(int j = 0; j < size - i - 1; ++j) {
                if(*(array + j) > *(array + j + 1)) {
                    int temp = *(array + j);
                    *(array + j) = *(array + j + 1);
                    *(array + j + 1) = temp;
                }
            }
        }

        // Print the sorted array
        printf("The sorted elements of the array are: ");
        for(int i = 0; i < size; ++i) {
            printf("%d", *(array + i));
            if(i < size - 1) {
                printf(", ");
            }
        }
        printf("\n");
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 5
Memory successfully allocated using malloc.
Enter element 1: 1
Enter element 2: 4
Enter element 3: 2
Enter element 4: 6
Enter element 5: 3
The elements of the array are: 1, 4, 2, 6, 3
The sorted elements of the array are: 1, 2, 3, 4, 6
```

- With Function

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

void getArray(int array[], int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &array[i]);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void printArray(int array[], int size) {
    for(int i = 0; i < size; ++i) {
        printf("%d", array[i]);
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

void bubbleSort(int array[], int size) {
    for(int i = 0; i < size - 1; ++i) {
        for(int j = 0; j < size - i - 1; ++j) {
            if(array[j] > array[j + 1]) {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printf("The elements of the array are: ");
        printArray(array, size);

        // Sort the array using Bubble Sort
        bubbleSort(array, size);

        // Print the sorted array
        printf("The sorted elements of the array are: ");
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 2
Memory successfully allocated using malloc.
Enter element 1: 4
Enter element 2: 1
The elements of the array are: 4, 1
The sorted elements of the array are: 1, 4
```

- With function and Pointer

dsa.c

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

void getArray(int *array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", array + i);
        // Flush the input buffer
        while (getchar() != '\n');
    }
}

void printArray(int *array, int size) {
    for(int i = 0; i < size; ++i) {
        printf("%d", *(array + i));
        if(i < size - 1) {
            printf(", ");
        }
    }
    printf("\n");
}

void bubbleSort(int *array, int size) {
    for(int i = 0; i < size - 1; ++i) {
        for(int j = 0; j < size - i - 1; ++j) {
            if(*(array + j) > *(array + j + 1)) {
                int temp = *(array + j);
                *(array + j) = *(array + j + 1);
                *(array + j + 1) = temp;
            }
        }
    }
}

int main() {
    int size;

    printf("Enter size of the array: ");
    scanf("%d", &size);
    // Flush the input buffer
    while (getchar() != '\n');

    // Allocate memory for the array
    int *array = (int*) malloc(size * sizeof(int));

    if(array == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else {
        printf("Memory successfully allocated using malloc.\n");

        // Get elements of the array from the user
        getArray(array, size);

        // Print the array elements
        printf("The elements of the array are: ");
        printArray(array, size);

        // Sort the array using Bubble Sort
        bubbleSort(array, size);

        // Print the sorted array
        printf("The sorted elements of the array are: ");
        printArray(array, size);
    }

    // Free the allocated memory
    free(array);

    return 0;
}
```

- Output

```
Enter size of the array: 4
Memory successfully allocated using malloc.
Enter element 1: 5
Enter element 2: 2
Enter element 3: 6
Enter element 4: 3
The elements of the array are: 5, 2, 6, 3
The sorted elements of the array are: 2, 3, 5, 6
```

- Singular Linked List
  - Traversing

dsa.c > main()

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

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if(newNode == NULL) {
        printf("Error! memory not allocated.");
        exit(0);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to add a node at the end of the list
void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if(*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while(temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

// Function to print the list
void printList(Node* head) {
    Node* temp = head;
    while(temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    Node* head = NULL;

    addNode(&head, 1);
    addNode(&head, 2);
    addNode(&head, 3);
    addNode(&head, 4);
    addNode(&head, 5);

    printList(head);

    return 0;
}
```

OUTPUT

```
Enter the number of nodes: 5
Enter data for node 1: 1
Enter data for node 2: 2
Enter data for node 3: 4
Enter data for node 4: 5
Enter data for node 5: 3
1 -> 2 -> 4 -> 5 -> 3 -> NULL
```

- Insertion

```c
Node* createNode(int data) {
        printf("Error! memory not allocated.");
        exit(0);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to add a node at the end of the list
void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if(*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while(temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

// Function to insert a node at a specific position
void insertNode(Node** head, int data, int position) {
    Node* newNode = createNode(data);
    if(position == 0) {
        newNode->next = *head;
        *head = newNode;
        return;
    }
    Node* temp = *head;
    for(int i = 0; i < position - 1; i++) {
        if(temp != NULL) {
            temp = temp->next;
        }
    }
    if(temp != NULL) {
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

// Function to print the list
void printList(Node* head) {
    Node* temp = head;
    while(temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    Node* head = NULL;
    int size, data, position;

    printf("Enter the number of nodes: ");
    scanf("%d", &size);

    for(int i = 0; i < size; i++) {
        printf("Enter data for node %d: ", i + 1);
        scanf("%d", &data);
        addNode(&head, data);
    }

    printf("Enter data and position for new node: ");
    scanf("%d %d", &data, &position);
    insertNode(&head, data, position);

    printList(head);

    return 0;
}
```

- Output

```
Enter the number of nodes: 4
Enter data for node 1: 2
Enter data for node 2: 1
Enter data for node 3: 4
Enter data for node 4: 5
Enter data and position for new node: 2

4
2 -> 1 -> 4 -> 5 -> 2 -> NULL
```

- Deletion

dsa.c > main()

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

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if(newNode == NULL) {
        printf("Error! memory not allocated.");
        exit(0);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to add a node at the end of the list
void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if(*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while(temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

// Function to insert a node at a specific position
void insertNode(Node** head, int data, int position) {
    Node* newNode = createNode(data);
    if(position == 0) {
        newNode->next = *head;
        *head = newNode;
        return;
    }
    Node* temp = *head;
    for(int i = 0; i < position - 1; i++) {
        if(temp != NULL) {
            temp = temp->next;
        }
    }
    if(temp != NULL) {
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

// Function to delete a node at a specific position
void deleteNode(Node** head, int position) {
    if(*head == NULL) {
        return;
    }
    Node* temp = *head;
    if(position == 0) {
        *head = temp->next;
        free(temp);
        return;
    }
    for(int i = 0; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }
    if(temp == NULL || temp->next == NULL) {
        return;
    }
    Node* next = temp->next->next;
    free(temp->next);
    temp->next = next;
```

- Output

```
Enter the number of nodes: 5
Enter data for node 1: 1
Enter data for node 2: 2
Enter data for node 3: 1
Enter data for node 4: 4
Enter data for node 5: 2
Enter data and position for new node: 1 2
Enter position of node to delete: 1
1 -> 1 -> 1 -> 4 -> 2 -> NULL
```

- Circular linked list
- Traversing

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

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if(newNode == NULL) {
        printf("Error! memory not allocated.");
        exit(0);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to add a node at the end of the list
void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if(*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while(temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

// Function to insert a node at a specific position
void insertNode(Node** head, int data, int position) {
    Node* newNode = createNode(data);
    if(position == 0) {
        newNode->next = *head;
        *head = newNode;
        return;
    }
    Node* temp = *head;
    for(int i = 0; i < position - 1; i++) {
        if(temp != NULL) {
            temp = temp->next;
        }
    }
    if(temp != NULL) {
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

// Function to delete a node at a specific position
void deleteNode(Node** head, int position) {
    if(*head == NULL) {
        return;
    }
    Node* temp = *head;
    if(position == 0) {
        *head = temp->next;
        free(temp);
        return;
    }
    for(int i = 0; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }
    if(temp == NULL || temp->next == NULL) {
        return;
    }
    Node* next = temp->next->next;
    free(temp->next);
    temp->next = next;
```

- Output

```
Enter the number of nodes: 5
Enter data for node 1: 2
Enter data for node 2:
1
Enter data for node 3: 2
Enter data for node 4: 2
Enter data for node 5: 1
Enter data and position for new node:
4

4
Enter position of node to delete: 4
2 -> 1 -> 2 -> 2 -> 1 -> NULL
```

- INSERTION

dsa.c > main()

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

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    if(newNode == NULL) {
        printf("Error! memory not allocated.");
        exit(0);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to add a node at the end of the list
void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if(*head == NULL) {
        *head = newNode;
        newNode->next = newNode; // Point next of new node to itself
    } else {
        Node* temp = *head;
        while(temp->next != *head) { // Traverse till the last node
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = *head; // Point next of new node to head
    }
}

// Function to insert a node at a specific position
void insertNode(Node** head, int data, int position) {
    Node* newNode = createNode(data);
    if(position == 0) {
        if(*head == NULL) {
            newNode->next = newNode; // Point next of new node to itself
            *head = newNode;
        } else {
            Node* temp = *head;
            while(temp->next != *head) { // Traverse till the last node
                temp = temp->next;
            }
            newNode->next = *head;
            temp->next = newNode;
            *head = newNode;
        }
        return;
    }
    Node* temp = *head;
    for(int i = 0; i < position - 1; i++) {
        if(temp != NULL) {
            temp = temp->next;
        }
    }
    if(temp != NULL) {
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

// Function to print the list
void printList(Node* head) {
    if(head != NULL) {
        Node* temp = head;
        do {
            printf("%d -> ", temp->data);
            temp = temp->next;
        } while(temp != head);
```

- Output

```
Enter the number of nodes: 4
Enter data for node 1: 4
Enter data for node 2: 2
Enter data for node 3: 3
Enter data for node 4: 1
Enter data and position for new node: 4
1
4 -> 4 -> 2 -> 3 -> 1 ->  (back to 4)
```

- DELELTION

```c
void insertNode(Node** head, int data, int position) {
        temp->next = newNode;
    }
}

// Function to delete a node at a specific position
void deleteNode(Node** head, int position) {
    if(*head == NULL) {
        printf("List is empty.\n");
        return;
    }
    Node* temp = *head;
    if(position == 0) {
        while(temp->next != *head) { // Traverse till the last node
            temp = temp->next;
        }
        if(*head == (*head)->next) { // If only one node
            free(*head);
            *head = NULL;
        } else {
            Node* temp2 = *head;
            temp->next = (*head)->next;
            *head = (*head)->next;
            free(temp2);
        }
        return;
    }
    for(int i = 0; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }
    if(temp != NULL) {
        Node* next = temp->next;
        temp->next = next->next;
        free(next);
    }
}

// Function to print the list
void printList(Node* head) {
    if(head != NULL) {
        Node* temp = head;
        do {
            printf("%d -> ", temp->data);
            temp = temp->next;
        } while(temp != head);
        printf(" (back to %d)\n", head->data);
    }
}

int main() {
    Node* head = NULL;
    int size, data, position;

    printf("Enter the number of nodes: ");
    scanf("%d", &size);

    for(int i = 0; i < size; i++) {
        printf("Enter data for node %d: ", i + 1);
        scanf("%d", &data);
        addNode(&head, data);
    }

    printf("Enter data and position for new node: ");
    scanf("%d %d", &data, &position);
    insertNode(&head, data, position);

    printf("Enter position of node to delete: ");
    scanf("%d", &position);
    deleteNode(&head, position);

    printList(head);

    return 0;
}
```

- Output

```
Enter the number of nodes: 4
Enter data for node 1: 1
Enter data for node 2:
12
Enter data for node 3:
1
Enter data for node 4:
1
Enter data and position for new node: 2
1
Enter position of node to delete: 2
1 -> 2 -> 1 -> 1 ->  (back to 1)
```