

Lab Report

Subject: Data Structure
Lab Report

Student Information

Name: Mahdi Hasan Shuo
Student ID: 251-115-030
Department: CSE
Batch: 62 (A)

Course Teacher:
Bushra Azmat Hussain
Metropolitan University

Submission Date: 29-11-2025

Signature: _____

*Metropolitan University
Sylhet, Bangladesh*

Question:

1. Insert an element at the beginning of an array

Code :

```
#include <stdio.h>

int main() {

    int insarry[100], n, i, x;

    printf("Enter size of array: ");

    scanf("%d", &n);

    printf("Enter array elements: ");

    for(i = 0; i < n; i++)

        scanf("%d", &insarry[i]);

    printf("Enter element to insert at beginning: ");

    scanf("%d", &x);

    for(i = n; i > 0; i--)

        insarry[i] = insarry[i - 1];

    insarry[0] = x;

    n++;

    printf("Array after insertion: ");

    for(i = 0; i < n; i++)

        printf("%d ", insarry[i]);

}
```

Output:

```
Output
Enter size of array: 3
Enter array elements: 3 4 5
Enter element to insert at beginning: 2
Array after insertion: 2 3 4 5

==== Code Execution Successful ====
```

2. Insert multiple elements at the beginning of an array

Code :

```
#include <stdio.h>

int main() {
    int a[200], extra[100];
    int n, m, i;
    printf("Enter size array: ");
    scanf("%d", &n);
    printf("Enter array element: ");
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("How many elements to insert : ");
    scanf("%d", &m);
    printf("Enter %d elements: ", m);
    for(i = 0; i < m; i++)
        scanf("%d", &extra[i]);
    for(i = n - 1; i >= 0; i--)
        a[i + m] = a[i];
    for(i = 0; i < m; i++)
        a[i] = extra[i];
    n = n + m;
}
```

```
printf("Array after insertion: ");

for(i = 0; i < n; i++)

    printf("%d ", a[i]);

}
```

Output:

Output

```
Enter size array: 3
Enter array element: 4 4 6
How many elements to insert : 3
Enter 3 elements: 1 4 7
Array after insertion: 1 4 7 4 4 6

==== Code Execution Successful ===
```

3 . Insert an element at the end of an array

Code :

```
#include <stdio.h>

int main() {

    int a[100], n, x, i;

    printf("Enter size of the array: ");

    scanf("%d", &n);

    printf("Enter element: ");

    for(i = 0; i < n; i++)

        scanf("%d", &a[i]);
```

```

printf("Enter element to insert : ");

scanf("%d", &x);

a[n] = x;

n++;

printf("Array after insert: ");

for(i = 0; i < n; i++)

printf("%d ", a[i]);

}

```

Output:

```

Output
Enter size of the array: 2
Enter element: 3 0
Enter element to insert : 1
Array after insert: 3 0 1
== Code Execution Successful ==

```

4. Insert multiple elements at the end of an array

Code :

```

#include <stdio.h>

int main() {

int a[200], insnum[100];

int n, m, i;

printf("Enter array size : ");

scanf("%d", &n);

printf("Enter Your array: ");

for(i = 0; i < n; i++)

```

```

scanf("%d", &a[i]);

printf("How many elements are insert : ");

scanf("%d", &m);

printf("Enter elements : ");

for(i = 0; i < m; i++)

scanf("%d", &insnum[i]);

for(i = 0; i < m; i++)

a[n + i] = insnum[i];

n += m;

printf("Array after insertion: ");

for(i = 0; i < n; i++)

printf("%d ", a[i]);

```

OutPut:

```

Output

Enter array size : 4
Enter Your array: 1 1 5 0
How many elements are insert : 2
Enter elements : 3 0
Array after insertion: 1 1 5 0 3 0

==== Code Execution Successful ====

```

5. Insert an element at a given position by use of an array.

Code :

```

#include <stdio.h>

int main() {

```

```

int a[100], n, inppos, x, i;

printf("Enter size: ");

scanf("%d", &n);

printf("Enter array: ");

for(i = 0; i < n; i++)

scanf("%d", &a[i]);

printf("Enter position (0-based index): ");

scanf("%d", &inppos);

printf("Enter element: ");

scanf("%d", &x);

for(i = n; i > inppos; i--)

a[i] = a[i - 1];

a[inppos] = x;

n++;

printf("Array after insertion: ");

for(i = 0; i < n; i++)

printf("%d ", a[i]);

}

```

Output:

Output
<pre> Enter size: 3 Enter array: 1 2 3 Enter position (0-based index): 2 Enter element: 8 Array after insertion: 1 2 8 3 ==== Code Execution Successful ==== </pre>

6. Insert multiple elements at a given position by use of an array.

Code :

```
#include <stdio.h>

int main() {
    int a[200], input_element[100];
    int n, m, pos, i;
    printf("Enter size: ");
    scanf("%d", &n);
    printf("Enter array: ");
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("Enter position: ");
    scanf("%d", &pos);
    printf("How many elements to insert: ");
    scanf("%d", &m);
    printf("Enter elements: ");
    for(i = 0; i < m; i++)
        scanf("%d", &input_element[i]);
    for(i = n - 1; i >= pos; i--)
        a[i + m] = a[i];
    for(i = 0; i < m; i++)
        a[pos + i] = input_element[i];
    n += m;
    printf("Array after insertion: ");
}
```

```
for(i = 0; i < n; i++)  
    printf("%d ", a[i]);  
}
```

Output:

```
Output  
Enter size: 3  
Enter array: 1 2 3  
Enter position: 2  
How many elements to insert: 1  
Enter elements: 3  
Array after insertion: 1 2 3 3  
== Code Execution Successful ==
```

7. Delete an element from the beginning of an array.

Code :

```
#include <stdio.h>  
  
int main() {  
    int a[100], n, i;  
    printf("Enter size : ");  
    scanf("%d", &n);  
    printf("Enter array elements: ");  
    for(i=0; i<n; i++)  
        scanf("%d", &a[i]);
```

```

for(i=0; i<n-1; i++)
    a[i] = a[i+1];
n--;
printf("Array after delete: ");
for(i=0; i<n; i++)
    printf("%d ", a[i]);
}

```

Output:

Output
<pre> Enter size : 3 Enter array elements: 1 3 4 Array after delete: 3 4 == Code Execution Successful == </pre>

8. Delete multiple elements from the beginning of an array.

Code:

```

#include <stdio.h>

int main() {
    int a[100], n, k, i;
    printf("Enter array size : ");
    scanf("%d", &n);
    printf("Enter array: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);

```

```

printf("How many elements to delete from beginning: ");

scanf("%d", &k);

for(i=0; i<n-k; i++)
    a[i] = a[i+k];

n -= k;

printf("Array after deletion: ");

for(i=0; i<n; i++)
    printf("%d ", a[i]);

}

```

Output:

```

Output

Enter array size : 4
Enter array: 1 4 5 7
How many elements to delete from beginning: 2
Array after deletion: 5 7

==== Code Execution Successful ====

```

9. Delete an element from the end of an array.

Code :

```

#include <stdio.h>

int main() {
    int a[100], n, i;
    printf("Enter size: ");

```

```

scanf("%d", &n);

printf("Enter array: ");

for(i=0; i<n; i++)

scanf("%d", &a[i]);

n--;

printf("Array after delet: ");

for(i=0; i<n; i++)

printf("%d ", a[i]);

}

```

Output:

```

Output

Enter size: 3
Enter array: 1 3 4
Array after delet: 1 3

==== Code Execution Successful ====

```

10. Delete multiple elements from the end of an array.

Code :

```

#include <stdio.h>

int main() {

int a[100], n, k, i;

printf("Enter size: ");

scanf("%d", &n);

printf("Enter array: ");

```

```

for(i=0; i<n; i++)
scanf("%d", &a[i]);

printf("How many elements to delete from this ary end: ");

scanf("%d", &k);

n -= k;

printf("Array after delet: ");

for(i=0; i<n; i++)
printf("%d ", a[i]);

}

```

Output:

Output
<pre> Enter size: 3 Enter array: 23 4 5 How many elements to delete from this ary end: 2 Array after delet: 23 ==== Code Execution Successful ==== </pre>

11. Delete an element from a given position by using an array.

Code :

```

#include <stdio.h>

int main() {

int a[100], n, posion, i;

printf("Enter size: ");

scanf("%d", &n);

```

```

printf("Enter array: ");

for(i=0; i<n; i++)
    scanf("%d", &a[i]);

printf("Enter delete position : ");

scanf("%d", &posion);

for(i=posion; i<n-1; i++)
    a[i] = a[i+1];

n--;

printf("Array after delete: ");

for(i=0; i<n; i++)
    printf("%d ", a[i]);

}

```

Output:

Output
<pre> Enter size: 3 Enter array: 4 5 2 Enter delete position : 1 Array after delete: 4 2 == Code Execution Successful == </pre>

12. Delete multiple elements from a given position by use of an array.

COde :

```

#include <stdio.h>

int main() {
    int a[100], n, pos, k, i;

```

```

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

printf("Enter array: ");
for(i=0; i<n; i++)
    scanf("%d", &a[i]);

printf("Enter the position: ");
scanf("%d", &pos);

printf("Number of elements to delete: ");
scanf("%d", &k);

for(i=pos; i<n-k; i++)
    a[i] = a[i+k];

n -= k;

printf("Array after deletion: ");
for(i=0; i<n; i++)
    printf("%d ", a[i]);
}

```

Output:

Output
<pre> Enter size: 5 Enter array: 2 4 5 7 8 Enter the position: 3 Number of elements to delete: 2 Array after deletion: 2 4 5 ==== Code Execution Successful ==== </pre>

13. Read an array from user and a number. Then search for the number if it is available on the array using Linear search.

CCode :

```
#include <stdio.h>

int main() {
    int a[100], n, find, i, found = 0;
    printf("Enter size of array: ");
    scanf("%d", &n);
    printf("Enter elements: ");
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    printf("Enter number for search: ");
    scanf("%d", &find);
    for(i = 0; i < n; i++) {
        if(a[i] == find) {
            found = 1;
            break;
        }
    }
    if(found)
        printf("Element found at index %d\n", i);
    else
        printf("Element not found.\n");
}
```

Output:

Output
Enter size of array: 4 Enter elements: 3 5 6 7 Enter number for search: 3 Element found at index 0 == Code Execution Successful ==

14. Read an array from user and a number. Then search the number if it is available on the array using Binary Search.

Code:

```
#include <stdio.h>

int main() {

    int a[100], n, key, low, high, mid, i;

    printf("Enter size of array: ");

    scanf("%d", &n);

    printf("Enter elements (sorted): ");

    for(i = 0; i < n; i++)

        scanf("%d", &a[i]);

    printf("Enter number to search: ");

    scanf("%d", &key);

    low = 0;

    high = n - 1;

    while(low <= high) {

        mid = (low + high) / 2;

        if(a[mid] == key) {
```

```

printf("Element found at index %d\n", mid);

return 0; }

else if(a[mid] < key)

low = mid + 1;

else

high = mid - 1; }

printf("Element not found.\n");

```

Output:

Output
<pre> Enter size of array: 3 Enter elements (sorted): 1 2 3 Enter number to search: 2 Element found at index 1 ==== Code Execution Successful ==== </pre>

15. Read an array from user and sort it using Bubble Sort in both ascending and descending order.

```

#include <stdio.h>

int main() {

int a[100], n, i, j, temp;

printf("Enter size: ");

scanf("%d", &n);

printf("Enter elements: ");

for(i=0; i<n; i++)

scanf("%d", &a[i]);

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

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

```

```

if(a[j] > a[j+1]) {
    temp = a[j];
    a[j] = a[j+1];
    a[j+1] = temp;
}

printf("\nAscending: ");

for(i=0; i<n; i++) {
    printf("%d ", a[i]);
}

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

    printf("\nDescending: ");

    for(i=0; i<n; i++) {
        printf("%d ", a[i]);
    }
}

```

Output:

```
Output
Enter size: 4
Enter elements: 1 2 3 4

Ascending: 1 2 3 4
Descending: 4 3 2 1

==== Code Execution Successful ====
```

16. Read an array from user and sort it using Selection Sort in both ascending and descending order.

Code :

```
#include <stdio.h>

int main() {
    int a[100], n, i, j, min, max, temp;
    printf("Enter size: ");
    scanf("%d", &n);
    printf("Enter elements: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    for(i=0; i<n-1; i++) {
        min = i;
        for(j=i+1; j<n; j++)
            if(a[j] < a[min])
                min = j;
        temp = a[i];
        a[i] = a[min];
        a[min] = temp;
    }
}
```

```

a[i] = a[min];
a[min] = temp;
printf("\nAscending: ");
for(i=0; i<n; i++)
printf("%d ", a[i]);
for(i=0; i<n-1; i++) {
    max = i;
    for(j=i+1; j<n; j++)
        if(a[j] > a[max])
            max = j;
    temp = a[i];
    a[i] = a[max];
    a[max] = temp;
}
printf("\nDescending: ");
for(i=0; i<n; i++)
printf("%d ", a[i]);

```

Output:

Output
<pre> Enter size: 5 Enter elements: 1 2 3 4 5 Ascending: 1 2 3 4 5 Descending: 5 4 3 2 1 ==== Code Execution Successful ==== </pre>

17. Read an array from the user and sort it using Insertion Sort in both ascending and descending order.

Code :

```
#include <stdio.h>

int main() {
    int a[100], n, i, j, key;
    printf("Enter size: ");
    scanf("%d", &n);
    printf("Enter elements: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    for(i=1; i<n; i++) {
        key = a[i];
        j = i - 1;
        while(j >= 0 && a[j] > key) {
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = key;
    }
    printf("\nAscending: ");
    for(i=0; i<n; i++)
        printf("%d ", a[i]);
    for(i=1; i<n; i++) {
        key = a[i];
        j = i - 1;
        while(j >= 0 && a[j] < key) {
```

```

a[j+1] = a[j];
j--;
a[j+1] = key; }

printf("\nDescending: ");

for(i=0; i<n; i++)
printf("%d ", a[i]);

}

```

Output:

```

Output
Enter size: 5
Enter elements: 1 2 5 6 7

Ascending: 1 2 5 6 7
Descending: 7 6 5 2 1

== Code Execution Successful ==

```

18. Read an array from the user and sort it using Merge Sort in both ascending and descending order.

Code :

```

#include <stdio.h>

void merge(int arr[], int l, int mid, int r, int asc) {

int n1 = mid - l + 1;
int n2 = r - mid;
int L[100], R[100];
int i, j, k;
for(i=0; i<n1; i++)
L[i] = arr[l + i];
for(j=0; j<n2; j++)

```

```

R[j] = arr[mid + 1 + j];
i = j = 0;
k = l;
while(i < n1 && j < n2) {
    if((asc && L[i] <= R[j]) || (!asc && L[i] >= R[j])) {
        arr[k++] = L[i++];
    } else {
        arr[k++] = R[j++];
    }
    while(i < n1)
        arr[k++] = L[i++];
    while(j < n2)
        arr[k++] = R[j++];
}
void mergeSort(int arr[], int l, int r, int asc) {
    if(l < r) {
        int mid = (l + r) / 2;
        mergeSort(arr, l, mid, asc);
        mergeSort(arr, mid+1, r, asc);
        merge(arr, l, mid, r, asc);
    }
}
void merge(int arr[], int l, int mid, int r, int asc);
void mergeSort(int arr[], int l, int r, int asc);
int main() {
    int a[100], n, i;
    printf("Enter size: ");
    scanf("%d", &n);
    printf("Enter elements: ");
}

```

```

for(i=0; i<n; i++)
    scanf("%d", &a[i]);
mergeSort(a, 0, n-1, 1);
printf("\nAscending: ");
for(i=0; i<n; i++)
    printf("%d ", a[i]);
mergeSort(a, 0, n-1, 0);
printf("\nDescending: ");
for(i=0; i<n; i++)
    printf("%d ", a[i]);
}

```

Output:

Output
<pre> Enter size: 4 Enter elements: 1 2 3 5 Ascending: 1 2 3 5 Descending: 5 3 2 1 ==== Code Execution Successful ==== </pre>

19. Demonstrate all the operations of Stack (Push, Pop, Peek/Top, isFull, isEmpty, Display).

Code :

```

#include <stdio.h>

#define MAX 100

int stack[MAX];
int top = -1;

```

```

int isFull() {
    return top == MAX - 1;
}

int isEmpty() {
    return top == -1;
}

void push(int x) {
    if(isFull()) {
        printf("Stack is Full!\n");
    } else {
        top++;
        stack[top] = x;
        printf("%d pushed into stack.\n", x);
    }
}

void pop() {
    if(isEmpty()) {
        printf("Stack is Empty!\n");
    } else {
        printf("%d popped from stack.\n", stack[top]);
        top--;
    }
}

void peek() {
    if(isEmpty()) {
        printf("Stack is Empty!\n");
    } else {
        printf("Top element: %d\n", stack[top]);
    }
}

void display() {
    if(isEmpty()) {
        printf("Stack is Empty!\n");
    }
}

```

```
    } else {  
  
        int i;  
  
        printf("Stack elements: ");  
  
        for(i = 0; i <= top; i++)  
            printf("%d ", stack[i]);  
  
        printf("\n");}  
  
int main() {  
  
    int choice, value;  
  
    while(1) {  
  
        printf("\n--- Stack Menu ---\n");  
  
        printf("1. Push\n2. Pop\n3. Peek\n4. isFull\n5. isEmpty\n6. Display\n7. Exit\n");  
  
        printf("Enter choice: ");  
  
        scanf("%d", &choice);  
  
        switch(choice) {  
  
            case 1:  
  
                printf("Enter value to push: ");  
  
                scanf("%d", &value);  
  
                push(value);  
  
                break;  
  
            case 2:  
  
                pop();  
  
                break;  
  
            case 3:  
  
                peek();  
  
                break;  
        }  
    }  
}
```

```
case 4:  
    if(isFull()) printf("Stack is Full.\n");  
    else printf("Stack is Not Full.\n");  
    break;  
  
case 5:  
    if(isEmpty()) printf("Stack is Empty.\n");  
    else printf("Stack is Not Empty.\n");  
    break;  
  
case 6:  
    display();  
    break;  
  
case 7:  
    return 0;  
  
default:  
    printf("Invalid choice!\n");  
}  
}
```

Output:

```
Output
--- Stack Menu ---
1. Push
2. Pop
3. Peek
4. isFull
5. isEmpty
6. Display
7. Exit
Enter choice: 1
Enter value to push: 4
4 pushed into stack.

--- Stack Menu ---
1. Push
2. Pop
3. Peek
4. isfull
5. isempty
6. Display
7. Exit
Enter choice: 6
Stack elements: 4

--- Stack Menu ---
1. Push
2. Pop
3. Peek
4. isFull
5. isEmpty
6. Display
7. Exit
Enter choice: |
```

20. Demonstrate all the operations of Linear Queue (Enqueue, Dequeue, Peek/Top, isFull, isEmpty, Display).

Code :

```
#include <stdio.h>

#define MAX 100

int queue[MAX];

int front = -1;

int rear = -1;

int isFull() {
```

```

return rear == MAX - 1; }

int isEmpty() {
    return front == -1 || front > rear; }

void enqueue(int x) {
    if(isFull()) {
        printf("Queue is Full!\n");
    } else {
        if(front == -1)
            front = 0;
        rear++;
        queue[rear] = x;
        printf("%d inserted into queue.\n", x);}}
}

void dequeue() {
    if(isEmpty()) {
        printf("Queue is Empty!\n");
    } else {
        printf("%d removed from queue.\n", queue[front]);
        front++;
        if(front > rear) {
            front = rear = -1; }}}

void peek() {
    if(isEmpty()) {
        printf("Queue is Empty!\n");
    } else {
        printf("Front element: %d\n", queue[front]);}}
}

```

```

void display() {
    if(isEmpty()) {
        printf("Queue is Empty!\n");
    } else {
        int i;
        printf("Queue elements: ");
        for(i = front; i <= rear; i++)
            printf("%d ", queue[i]);
        printf("\n");}
}

int main() {
    int choice, value;
    while(1) {
        printf("\n--- Linear Queue Menu ---\n");
        printf("1. Enqueue\n2. Dequeue\n3. Peek\n4. isFull\n5. isEmpty\n6. Display\n7. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
        }
    }
}

```

```
case 3:  
    peek();  
    break;  
  
case 4:  
    if(isFull()) printf("Queue is Full.\n");  
    else printf("Queue is Not Full.\n");  
    break;  
  
case 5:  
    if(isEmpty()) printf("Queue is Empty.\n");  
    else printf("Queue is Not Empty.\n");  
    break;  
  
case 6:  
    display();  
    break;  
  
case 7:  
    return 0;  
  
default:  
    printf("Invalid choice!\n");}}
```

Output:

```
Output Clear
--- Linear Queue Menu ---
1. Enqueue
2. Dequeue
3. Peek
4. isFull
5. isEmpty
6. Display
7. Exit
Enter choice: 1
Enter value to insert: 4
4 inserted into queue.

--- Linear Queue Menu ---
1. Enqueue
2. Dequeue
3. Peek
4. isFull
5. isEmpty
6. Display
7. Exit
Enter choice: 6
Queue elements: 4
```

21. Demonstrate all the operations of Circular Queue (Enqueue, Dequeue, Peek/Top, isFull, isEmpty, Display).

Code :

```
#include <stdio.h>

#define MAX 100

int cq[MAX];

int front = -1, rear = -1;

int isFull() {
    return (front == (rear + 1) % MAX);}
int isEmpty() {
    return (front == -1);}
void enqueue(int x) {
    if(isFull()) {
        printf("Circular Queue is Full!\n");
        return;
    }
    if(front == -1)
        front = 0;
    rear = (rear + 1) % MAX;
    cq[rear] = x;
}
```

```

if(front == -1)

    front = rear = 0;

else

    rear = (rear + 1) % MAX;

    cq[rear] = x;

    printf("%d inserted.\n", x);}

void dequeue() {

    if(isEmpty()) {

        printf("Circular Queue is Empty!\n");

        return; }

    printf("%d removed.\n", cq[front]);

    if(front == rear)

        front = rear = -1;

    else

        front = (front + 1) % MAX; }

void peek() {

    if(isEmpty())

        printf("Circular Queue is Empty!\n");

    else

        printf("Front: %d\n", cq[front]); }

void display() {

    if(isEmpty()) {

        printf("Circular Queue is Empty!\n");

        return; }

    int i = front;

```

```

printf("Circular Queue: ");

while(1) {

    printf("%d ", cq[i]);

    if(i == rear) break;

    i = (i + 1) % MAX;

    printf("\n");

}

int main() {

    int ch, val;

    while(1) {

        printf("\n1.Enqueue \n2.Dequeue \n3.Peek \n4.Display \n5.Exit\n");

        scanf("%d", &ch);

        if(ch == 1) {

            scanf("%d", &val);

            enqueue(val);

        }

        else if(ch == 2) dequeue();

        else if(ch == 3) peek();

        else if(ch == 4) display();

        else break;

    }

}

```

Output:

```
Output
1.Enqueue
2.Dequeue
3.Peek
4.Display
5.Exit
1
4
4 inserted.

1.Enqueue
2.Dequeue
3.Peek
4.Display
5.Exit
4
Circular Queue: 4
```

22. Read some data from user and create a Singly Linked List
23. Read a Singly Linked List and insert an element at the beginning.
24. Read a Singly Linked List and insert an element at the ending.
25. Read a Singly Linked List and insert an element at a given position.
26. Read a Singly Linked List and delete an element from the beginning.
27. Read a Singly Linked List and delete an element from the ending.
28. Read a Singly Linked List and delete an element from a given position.
29. Read a Singly Linked List and print it in reverse order.

Code:

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

struct Node {
    int data;
    struct Node *next;
};
```

```
struct Node *head = NULL;

void createList(int n) {

    int val;

    struct Node *newNode, *temp;

    for(int i=0; i<n; i++) {

        scanf("%d", &val);

        newNode = (struct Node*)malloc(sizeof(struct Node));

        newNode->data = val;

        newNode->next = NULL;

        if(head == NULL)

            head = newNode;

        else {

            temp = head;

            while(temp->next != NULL)

                temp = temp->next;

            temp->next = newNode;}}}

void insertBegin(int val) {

    struct Node *newNode = malloc(sizeof(struct Node));

    newNode->data = val;

    newNode->next = head;

    head = newNode;}

void insertEnd(int val) {

    struct Node *newNode = malloc(sizeof(struct Node));

    newNode->data = val;

    newNode->next = NULL;
```

```
if(head == NULL) head = newNode;

else {

    struct Node *temp = head;

    while(temp->next) temp = temp->next;

    temp->next = newNode; }

void insertPos(int pos, int val) {

    struct Node *newNode = malloc(sizeof(struct Node));

    newNode->data = val;

    if(pos == 0) {

        newNode->next = head;

        head = newNode;

        return; }

    struct Node *temp = head;

    for(int i=0; i<pos-1; i++)

        temp = temp->next;

    newNode->next = temp->next;

    temp->next = newNode; }

void deleteBegin() {

    if(head == NULL) return;

    struct Node *temp = head;

    head = head->next;

    free(temp); }

void deleteEnd() {

    if(head == NULL) return;

    if(head->next == NULL) {
```

```

free(head);

head = NULL;

return; }

struct Node *temp = head;

while(temp->next->next != NULL)

    temp = temp->next;

    free(temp->next);

    temp->next = NULL; }

void deletePos(int pos) {

if(pos == 0) {

deleteBegin();

return; }

struct Node *temp = head;

for(int i=0; i<pos-1; i++)

    temp = temp->next;

struct Node *del = temp->next;

temp->next = del->next;

free(del);}

void printList() {

struct Node *temp = head;

while(temp) {

printf("%d ", temp->data);

temp = temp->next; }

printf("\n"); }

void printReverse(struct Node *node) {

```

```
if(node == NULL) return;  
  
printReverse(node->next);  
  
printf("%d ", node->data);}  
  
int main() {  
  
    int n, val, pos;  
  
    // Create List Q22  
  
    scanf("%d", &n);  
  
    createList(n);  
  
    printList();  
  
    // Insert beginning Q23  
  
    scanf("%d", &val);  
  
    insertBegin(val);  
  
    printList();  
  
    // Insert end Q24  
  
    scanf("%d", &val);  
  
    insertEnd(val);  
  
    printList();  
  
    // Insert position Q25  
  
    scanf("%d %d", &pos, &val);  
  
    insertPos(pos, val);  
  
    printList();  
  
    // Delete beginning Q26  
  
    deleteBegin();  
  
    printList();  
  
    // Delete end Q27
```

```

deleteEnd();
printList();
// Delete position Q28
scanf("%d", &pos);
deletePos(pos);
printList();
// Print Reverse Q29
printReverse(head);
}

```

Output:

```

input
Enter the number of createList: 2
2 3
2 3

Enter the value to insert at the beginning: 1
1 2 3

Enter the value to insert at the end: 3
1 2 3 3

Enter the position and value to insert: 2
2 5
1 2 2 3 3
2 2 3 3

Enter the position to delete: 2 2 3

```

30. Read some data from user and create a Doubly Linked List

C0de:

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node *prev, *next;
};

struct Node* createDoublyList(int n) {

    struct Node *head = NULL, *temp, *newNode;

    int value;

    for(int i = 0; i < n; i++) {

        printf("Enter value: ");

        scanf("%d", &value);

        newNode = (struct Node*)malloc(sizeof(struct Node));

        newNode->data = value;

        newNode->prev = newNode->next = NULL;

        if(head == NULL) {

            head = newNode;

        } else {

            temp = head;

            while(temp->next != NULL)

                temp = temp->next;

            temp->next = newNode;

            newNode->prev = temp;
        }
    }

    return head;
}
```

```

void display(struct Node *head) {

    struct Node *temp = head;

    printf("List: ");

    while(temp != NULL) {

        printf("%d ", temp->data);

        temp = temp->next; }

    printf("\n"); }

int main() {

    int n;

    printf("How many nodes :   ");

    scanf("%d", &n);

    struct Node *head = createDoublyList(n);

    display(head);

    return 0; }

```

Output:

Output
<pre> How many nodes : 3 Enter value: 2 3 4 Enter value: Enter value: List: 2 3 4 *** Code Execution Successful *** </pre>

30. Read some data from user and create a Doubly Linked List

31. Read a Doubly Linked List and insert an element at the beginning.

32. Read a Doubly Linked List and insert an element at the ending.

33. Read a Doubly Linked List and insert an element at a given position.
34. Read a Doubly Linked List and delete an element from the beginning.
35. Read a Doubly Linked List and delete an element from the ending.
36. Read a Doubly Linked List and delete an element from a given position.

Code :

```
#include <stdio.h>
#include <stdlib.h>
struct DNode {
    int data;
    struct DNode *prev, *next;
};
struct DNode *head = NULL;
void createDList(int n) {
    int val;
    struct DNode *newNode, *temp;
    for(int i=0; i<n; i++) {
        scanf("%d", &val);
        newNode = malloc(sizeof(struct DNode));
        newNode->data = val;
        newNode->prev = NULL;
        newNode->next = NULL;
        if(head == NULL)
            head = newNode;
        else {
            temp = head;
            while(temp->next)
                temp = temp->next;
            temp->next = newNode;
            newNode->prev = temp;
        }
    }
}
void insertBeginD(int val) {
    struct DNode *newNode = malloc(sizeof(struct DNode));
    newNode->data = val;
    newNode->prev = NULL;
    newNode->next = head;
    if(head) head->prev = newNode;
    head = newNode;
}
void insertEndD(int val) {
    struct DNode *newNode = malloc(sizeof(struct DNode));
    newNode->data = val;
    newNode->next = NULL;
```

```

if(head == NULL) {
    newNode->prev = NULL;
    head = newNode;
} else {
    struct DNode *temp = head;
    while(temp->next) temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp; }
void insertPosD(int pos, int val) {
    struct DNode *newNode = malloc(sizeof(struct DNode));
    newNode->data = val;
    if(pos == 0) {
        insertBeginD(val);
        return; }
    struct DNode *temp = head;
    for(int i=0; i<pos-1; i++)
        temp = temp->next;
    newNode->next = temp->next;
    newNode->prev = temp;
    if(temp->next) temp->next->prev = newNode;
    temp->next = newNode; }
void deleteBeginD() {
    if(head == NULL) return;
    struct DNode *temp = head;
    head = head->next;
    if(head) head->prev = NULL;
    free(temp); }
void deleteEndD() {
    if(head == NULL) return;
    struct DNode *temp = head;
    while(temp->next) temp = temp->next;
    if(temp->prev)
        temp->prev->next = NULL;
    else
        head = NULL;
    free(temp); }
void deletePosD(int pos) {
    if(pos == 0) {
        deleteBeginD();
        return; }
    struct DNode *temp = head;
    for(int i=0; i<pos; i++)
        temp = temp->next;
    temp->prev->next = temp->next;
}

```

```

if(temp->next)
    temp->next->prev = temp->prev;
free(temp); }

void printDLi st() {
    struct DNode *temp = head;
    while(temp) {
        printf("%d ", temp->data);
        temp = temp->next; }
    printf("\n"); }

int main() {
    int n, val, pos;
    // Print Reverse Q30
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    createDLi st(n);
    printDLi st();
    // Print Reverse Q31
    printf("Enter value to insert at beginning: ");
    scanf("%d", &val);
    insertBeginD(val);
    printDLi st();
    // Print Reverse Q32
    printf("Enter value to insert at end: ");
    scanf("%d", &val);
    insertEndD(val);
    printDLi st();
    // Print Reverse Q33
    printf("Enter position and value to insert: ");
    scanf("%d %d", &pos, &val);
    insertPosD(pos, val);
    printDLi st();
    // Print Reverse Q34
    deleteBeginD();
    printDLi st();
    // Print Reverse Q35
    deleteEndD();
    printDLi st();
    // Print Reverse Q36
    printf("Enter position to delete: ");
    scanf("%d", &pos);
    deletePosD(pos);
    printDLi st(); }

}

```

OUTPUT:

```
Output
* Enter number of nodes: 3
1 2 3
1 2 3
Enter value to insert at beginning: 2
2 1 2 3
Enter value to insert at end: 5
2 1 2 3 5
Enter position and value to insert: 2
3
2 1 3 2 3 5
1 3 2 3 5
1 3 2 3
Enter position to delete:
```

37. Read some data from user and create a Circular Linked List.\

Code :

```
#include <stdio.h>
#include <stdlib.h>
struct CNode {
    int data;
    struct CNode *next;
};
struct CNode *last = NULL;
void createCList(int n) {
    int val;
    struct CNode *newNode;
    for(int i=0; i<n; i++) {
        scanf("%d", &val);
```

```

newNode = malloc(sizeof(struct CNode));
newNode->data = val;
if(last == NULL) {
    last = newNode;
    newNode->next = newNode;
}
else {
    newNode->next = last->next;
    last->next = newNode;
    last = newNode;
}}
void printCList() {
    if(last == NULL) return;
    struct CNode *temp = last->next;
    do {
        printf("%d ", temp->data);
        temp = temp->next;
    } while(temp != last->next);
    printf("\n");
}
int main() {
    int n;
    scanf("%d", &n);
    createCList(n);
    printCList();
}

```

OUTPUT:

3
2 3 4
2 3 4

38. Calculate the sum of numbers from 1 to n(given by user) using recursion.

Code :

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

```
int sum(int n) {
    if(n == 0) return 0;
    return n + sum(n - 1);
}

int main() {
    int n;
    scanf("%d", &n);
    printf("Sum = %d\n", sum(n));
}
```

OUTPUT:



```
input
4
Sum = 10
```

39. Print the Fibonacci Series using recursion.

Code :

```
#include <stdio.h>

int fib(int n) {
    if(n <= 1) return n;
    return fib(n-1) + fib(n-2);
}

int main() {
    int n;
    scanf("%d", &n);

    for(int i=0; i<n; i++)
        printf("%d ", fib(i));
}
```

OUTPUT:



A screenshot of a terminal window titled "input". The window shows the following text:
5
0 1 1 2 3
...Program finished with exit code 0

Thank You