Stack array

```
#include <stdio.h>

#include <stdlib.h>

void push(int stack[], int n);

void pop(int stack[]);

void display(int stack[]);


int top=-1;

int main()

{

   int n;

   printf("Enter stack size: ");

   scanf("%d", &n);

   int stack[n];

   push(stack, n);

   display(stack);

   pop(stack);

   display(stack);


   return 0;

}


void push(int stack[], int n)

{

   int max_stack=n, num;

   if(top==max_stack-1)

   {

      printf("Overflow\n");

   }
```

```
    else
    {
        printf("Enter number you want to push: \n");
        for(int i=1; i<=max_stack; i++){
        scanf("%d", &num);
        top++;
        stack[top]=num;}
    }
}

void pop(int stack[])
{
    if(top==-1)
    {
        printf("Underflow\n");
    }
    else
    {
        printf("After popped %d\n", stack[top]);
        top--;
    }
}

void display(int stack[])
{
    if(top==-1)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("The stack is: \n");
        for(int i=0; i<=top; i++)
        {
            printf("%d  ", stack[i]);
        }
    }
    printf("\nThe top value is: %d\n", stack[top]);
}
```

Queue array

```
#include <stdio.h>
#include <stdlib.h>
void enqueue(int queues[], int n);
void dequeue(int queues[]);
void display();

int front=-1, rear=-1;
int main()
{
    int n;
    printf("Enter Queue size: ");
    scanf("%d", &n);
    int queues[n];
    enqueue(queues, n);
    display(queues);
    dequeue(queues);
    display(queues);
    return 0;
}

void enqueue(int queues[], int n)
{
    int num;
    for(int i=0; i<n; i++)
    {
        if(rear==n-1)
        {
            printf("Overflow\n");
            break;
        }
        else if(front == -1 && rear==-1)
        {
            front=0;
            rear=0;
            printf("Enter number: ");
            scanf("%d", &num);
            queues[rear]=num;
        }
        else
        {

            rear++;
            printf("Enter Number: ");
            scanf("%d", &num);
            queues[rear]=num;
        }
    }
}

void dequeue(int queues[])
{
    if(front==-1 && rear==-1)
    {
        printf("Queue is empty\n");
    }
    else if(front == rear)
    {
        printf("\n%d  ", queues[front]);
        front=-1;
        rear=-1;
    }
    else
    {
        printf("\n%d\n", queues[front]);
        front++;
    }
}

void display(int queues[])
{
    if(front == -1 && rear==-1)
    {
        printf("\nQueues is empty.\n");
    }
    else
```

```
  {
    printf("\nThe Queues value is: \n");
    for(int i=front; i<=rear; i++)
    {
      printf("%d ", queues[i]);
    }
  }
}
```

Queue linked list

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

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

struct Node* top = NULL;

void push(int num);
void pop();
void display();

int main() {
  int n;
  printf("Enter stack size: ");
  scanf("%d", &n);

  for (int i = 1; i <= n; i++) {
      int num;
      printf("Enter number you want to push: ");
      scanf("%d", &num);
      push(num);
  }

  display();

  pop();

  display();

  return 0;
}

void push(int num) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  if (newNode == NULL) {
      printf("Memory allocation failed.\n");
      exit(1);
  }

  newNode->data = num;
  newNode->next = top;
  top = newNode;
}

void pop() {
  if (top == NULL) {
```

```c
        printf("Underflow\n");
        return;
    }

    struct Node* temp = top;
    printf("After popped %d\n", temp->data);
    top = top->next;
    free(temp);
}

void display() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }

    struct Node* current = top;
    printf("The stack is: \n");

    while (current != NULL) {
        printf("%d  ", current->data);
        current = current->next;
    }

    printf("\nThe top value is: %d\n", top->data);
}
```

Queue linked list

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int data;

   struct Node* next;

};


struct Node* front = NULL;

struct Node* rear = NULL;


void enqueue(int num);

void dequeue();

void display();


int main() {

   int n;

   printf("Enter Queue size: ");

   scanf("%d", &n);


   for (int i = 1; i <= n; i++) {

      int num;

      printf("Enter number: ");

      scanf("%d", &num);

      enqueue(num);

   }


   display();
```

```
    dequeue();

    display();

    return 0;
}

void enqueue(int num) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }

    newNode->data = num;
    newNode->next = NULL;

    if (front == NULL) {
        front = newNode;
        rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
}

void dequeue() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }

    struct Node* temp = front;
    printf("Dequeued: %d\n", temp->data);

    if (front == rear) {
        front = NULL;
        rear = NULL;
    } else {
        front = front->next;
    }

    free(temp);
}

void display() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }

    struct Node* current = front;
    printf("The Queue is: \n");

    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }

    printf("\n");
}
```