INPUT

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int data;
   struct Node *next;
   struct Node *prev;
};
struct Node *createNode(int data)
{
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}
void insertAtFront(struct Node **head, int data)
{
   struct Node *newNode = createNode(data);
   if (*head == NULL)
   {
      *head = newNode;
   }
   else
   {
      newNode->next = *head;
      (*head)->prev = newNode;
      *head = newNode;
   }
}
```

```
void deleteFromEnd(struct Node **head)
{
    if (*head == NULL)
    {
        printf("The list is empty.\n");
        return;
    }
    struct Node *temp = *head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    if (temp->prev == NULL)
    {
        *head = NULL;
    }
    else
    {
        temp->prev->next = NULL;
    }
    free(temp);
}
void display(struct Node *head)
{
    if (head == NULL)
    {
        printf("The list is empty.\n");
        return;
    }
   struct Node *temp = head;
    printf("Doubly Linked List: ");
```

```
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main()
{
    struct Node *head = NULL;
    int choice, value;
    do
    {  printf("\nMenu:\n");
        printf("1. Insert at front\n");
        printf("2. Delete from end\n");
        printf("3. Display list\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter value to insert: ");
            scanf("%d", &value);
            insertAtFront(&head, value);
            break;
        case 2:
            deleteFromEnd(&head);
            break;
        case 3:
            display(head);
```

```
            break;
        case 4:
            exit;
        default:
            printf("Invalid choice! Please try again.\n");
        }} while (choice != 4);
    return 0;
}
```

OUTPUT

Insertion at front:

```
Menu:
1. Insert at front
2. Delete from end
3. Display list
4. Exit
Enter your choice: 3
Doubly Linked List: 69 74 16 65 23

Menu:
1. Insert at front
2. Delete from end
3. Display list
4. Exit
Enter your choice: 1
Enter value to insert: 31

Menu:
1. Insert at front
2. Delete from end
3. Display list
4. Exit
Enter your choice: 3
Doubly Linked List: 31 69 74 16 65 23
```

Deletion at last:

```
Menu:
1. Insert at front
2. Delete from end
3. Display list
4. Exit
Enter your choice: 2

Menu:
1. Insert at front
2. Delete from end
3. Display list
4. Exit
Enter your choice: 3
Doubly Linked List: 31 69 74 16 65
```

INPUT

```
#include <stdio.h>
#include <stdlib.h>
struct Node{
   int data;
   struct Node *next;
};
struct Node *createNode(int data)
{
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = newNode;
   return newNode;
}
void insertAtFront(struct Node **head, int data){
   struct Node *newNode = createNode(data);
   if (*head == NULL)
   {
      *head = newNode;
   }
   else{
      struct Node *temp = *head;
      while (temp->next != *head)
      {
         temp = temp->next;
      }
      temp->next = newNode;
   }
   newNode->next = *head;
   *head = newNode;
}
```

```
void deleteAtEnd(struct Node **head)
{
    if (*head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct Node *temp = *head;
    struct Node *prev = NULL;
    if (temp->next == *head)
    {
        free(temp);
        *head = NULL;
        return;
    }
    while (temp->next != *head)
    {
        prev = temp;
        temp = temp->next;
    }
    prev->next = *head;
    free(temp);
}
void display(struct Node *head)
{
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    struct Node *temp = head;
```

```
    do{
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("(head)\n");
}
int main()
{
    struct Node *head = NULL;
    int choice, data;
    do
    {  printf("\nMenu:\n");
        printf("1. Insertion at Front\n");
        printf("2. Deletion at End\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter data to insert at front: ");
            scanf("%d", &data);
            insertAtFront(&head, data);
            break;
        case 2:
            deleteAtEnd(&head);
            break;
        case 3:
            display(head);
            break;
```

```
        case 4:
            exit;
        default:
            printf("Invalid choice.\n");
        }} while (choice != 4);
    return 0;
}
```

OUTPUT

Insertion at front:

```
Menu:
1. Insertion at Front
2. Deletion at End
3. Display
4. Exit
Enter your choice: 3
17 -> 36 -> 54 -> 22 -> (head)

Menu:
1. Insertion at Front
2. Deletion at End
3. Display
4. Exit
Enter your choice: 1
Enter data to insert at front: 96

Menu:
1. Insertion at Front
2. Deletion at End
3. Display
4. Exit
Enter your choice: 3
96 -> 17 -> 36 -> 54 -> 22 -> (head)
```

Deletion at last:

```
Menu:
1. Insertion at Front
2. Deletion at End
3. Display
4. Exit
Enter your choice: 2

Menu:
1. Insertion at Front
2. Deletion at End
3. Display
4. Exit
Enter your choice: 3
96 -> 17 -> 36 -> 54 -> (head)
```

INPUT

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node *next;
};
struct Stack{
   struct Node *top;
};
struct Stack *createStack()
{
   struct Stack *stack = (struct Stack *)malloc(sizeof(struct Stack));
   stack->top = NULL;
   return stack;
}
int isEmpty(struct Stack *stack)
{
   return stack->top == NULL;
}
void push(struct Stack *stack, int data)
{
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = stack->top;
   stack->top = newNode;
}
int pop(struct Stack *stack)
{
```

```
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return -1;
    }
    struct Node *temp = stack->top;
    int poppedData = temp->data;
    stack->top = stack->top->next;
    free(temp);
    return poppedData;
}
int peek(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return -1;
    }
    return stack->top->data;
}
void display(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return;
    }
    struct Node *temp = stack->top;
    printf("Stack elements: ");
    while (temp != NULL)
    {
```

```
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main()
{
    struct Stack *stack = createStack();
    int choice, data;
    do   {
        printf("\nMenu:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter data to push: ");
            scanf("%d", &data);
            push(stack, data);
            break;
        case 2:
            data = pop(stack);
            if (data != -1)
            {
                printf("Popped: %d\n", data);
            }
```

```
            break;
        case 3:
            data = peek(stack);
            if (data != -1)
            {
                printf("Top element: %d\n", data);
            }
            break;
        case 4:
            display(stack);
            break;
        case 5:
            exit;
            break;
        default:
            printf("Invalid choice.\n");
        }
    } while (choice != 5);
    return 0;
}
```

OUTPUT

```
Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 4
Stack elements: 45 76 24 21

Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter data to push: 88
```

```
Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 3
Top element: 88
```

```
Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 2
Popped: 88

Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 4
Stack elements: 45 76 24 21
```

INPUT

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node *next;
};
struct Queue {
   struct Node *front;
   struct Node *rear;
};
struct Queue *createQueue()
{
   struct Queue *queue = (struct Queue *)malloc(sizeof(struct Queue));
   queue->front = queue->rear = NULL;
   return queue;
}
int isEmpty(struct Queue *queue)
{
   return queue->front == NULL;
}
void enqueue(struct Queue *queue, int data)
{
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   if (queue->rear == NULL)
   {
      queue->front = queue->rear = newNode;
      return;
```

```
    }
    queue->rear->next = newNode;
    queue->rear = newNode;
}

int dequeue(struct Queue *queue)
{
    if (isEmpty(queue))
    {
        printf("Queue is empty!\n");
        return -1;
    }
    struct Node *temp = queue->front;
    int dequeuedData = temp->data;
    queue->front = queue->front->next;
    if (queue->front == NULL)
    {
        queue->rear = NULL;
    }
    free(temp);
    return dequeuedData;
}
void display(struct Queue *queue)
{
    if (isEmpty(queue))
    {
        printf("Queue is empty!\n");
        return;
    }
    struct Node *temp = queue->front;
    printf("Queue elements: ");
```

```
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main()
{
    struct Queue *queue = createQueue();
    int choice, data;
    do {
        printf("\nMenu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter data to enqueue: ");
            scanf("%d", &data);
            enqueue(queue, data);
            break;
        case 2:
            data = dequeue(queue);
            if (data != -1)
            {
                printf("Dequeued: %d\n", data);
```

```
        }
        break;
    case 3:
        display(queue);
        break;
    case 4:
        break;
    default:
        printf("Invalid choice.\n");
    }
  } while (choice != 4);
  return 0;
}
```

OUTPUT

```
Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
Dequeued: 21

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue elements: 32 53 17
```

```
Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue elements: 21 32 53

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter data to enqueue: 17

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue elements: 21 32 53 17
```

INPUT

```
#include <stdio.h>
void bubbleSort(int arr[], int n)
{
   for (int i = 0; i < n - 1; i++)
   {
      for (int j = 0; j < n - i - 1; j++)
      {
         if (arr[j] > arr[j + 1])
         {
            // Swap arr[j] and arr[j+1]
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
         }
      }
   }
}
void insertionSort(int arr[], int n)
{
   for (int i = 1; i < n; i++)
   {
      int key = arr[i];
      int j = i - 1;
      while (j >= 0 && arr[j] > key)
      {
         arr[j + 1] = arr[j];
         j--;
      }
```

```
        arr[j + 1] = key;
    }
}
void displayArray(int arr[], int n)
{
    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main()
{
    int choice, n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    do
    {
        printf("\nMenu:\n");
        printf("1. Bubble Sort\n");
        printf("2. Insertion Sort\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
```

```
        scanf("%d", &choice);
        switch (choice)
        {       case 1:
            bubbleSort(arr, n);
            displayArray(arr, n);
            break;
        case 2:
            insertionSort(arr, n);
            displayArray(arr, n);
            break;
        case 3:
            exit;
        default:
            printf("Invalid choice.\n");
        }
    } while (choice != 3);
    return 0;
}
```

OUTPUT

```
Enter 4 elements:
1
5
3
8

Menu:
1. Bubble Sort
2. Insertion Sort
3. Exit
Enter your choice: 1
Sorted array: 1 3 5 8
```

```
real    0m9.541s
user    0m0.015s
sys     0m0.123s
```

```
Enter number of elements: 4
Enter 4 elements:
4
2
7
9

Menu:
1. Bubble Sort
2. Insertion Sort
3. Exit
Enter your choice: 2
Sorted array: 2 4 7 9
```

```
real    0m7.705s
user    0m0.046s
sys     0m0.109s
```

INPUT

```
#include <stdio.h>
void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++) {
        L[i] = arr[left + i];
    }
    for (j = 0; j < n2; j++) {
        R[j] = arr[mid + 1 + j];
    }
    i = 0;
    j = 0;
    k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
         else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
```

```
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}
void displayArray(int arr[], int n) {
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
```

```
    }
    mergeSort(arr, 0, n - 1);
    displayArray(arr, n);
    return 0;
}
```

OUTPUT

```
Enter number of elements: 5
Enter 5 elements:
3
7
2
5
9
Sorted array: 2 3 5 7 9

real    0m11.833s
user    0m0.045s
sys     0m0.092s
```