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

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

// Function prototypes
Node* createNode(int data);
Node* push(Node* top, int data);
Node* pop(Node* top, int* poppedValue);
Node* enqueue(Node* rear, int data);
Node* dequeue(Node* front, int* dequeuedValue);
void display(Node* head);

int main() {
    Node* stackTop = NULL;
    Node* queueFront = NULL;
    Node* queueRear = NULL;
    int choice, data, poppedValue, dequeuedValue;

    do {
        printf("\nMenu:\n");
        printf("1. Push (Stack)\n");
        printf("2. Pop (Stack)\n");
        printf("3. Enqueue (Queue)\n");
        printf("4. Dequeue (Queue)\n");
        printf("5. Display\n");
        printf("6. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to push: ");
                scanf("%d", &data);
                stackTop = push(stackTop, data);
                break;

            case 2:
                stackTop = pop(stackTop, &poppedValue);
```

```
            if (stackTop != NULL) {
                printf("Popped element: %d\n", poppedValue);
            }
            break;

        case 3:
            printf("Enter data to enqueue: ");
            scanf("%d", &data);
            queueRear = enqueue(queueRear, data);
            if (queueFront == NULL) {
                queueFront = queueRear;
            }
            break;

        case 4:
            queueFront = dequeue(queueFront, &dequeuedValue);
            if (queueFront != NULL) {
                printf("Dequeued element: %d\n", dequeuedValue);
                if (queueFront == NULL) {
                    queueRear = NULL; // Reset rear if queue becomes empty
                }
            }
            break;

        case 5:
            display(queueFront);
            break;

        case 6:
            printf("Exiting the program.\n");
            break;

        default:
            printf("Invalid choice. Please enter a valid option.\n");
      }

   } while (choice != 6);

   return 0;
}

// Function to create a new node with the given data
Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
```

```
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to push an element onto the stack
Node* push(Node* top, int data) {
    Node* newNode = createNode(data);
    newNode->next = top;
    return newNode;
}

// Function to pop an element from the stack
Node* pop(Node* top, int* poppedValue) {
    if (top == NULL) {
        printf("Stack underflow. Cannot pop from an empty stack.\n");
        return NULL;
    }
    *poppedValue = top->data;
    Node* temp = top;
    top = top->next;
    free(temp);
    return top;
}

// Function to enqueue an element into the queue
Node* enqueue(Node* rear, int data) {
    Node* newNode = createNode(data);
    if (rear == NULL) {
        // If the queue is empty, set both front and rear to the new node
        return newNode;
    }
    rear->next = newNode;
    return newNode;
}

// Function to dequeue an element from the queue
Node* dequeue(Node* front, int* dequeuedValue) {
    if (front == NULL) {
        printf("Queue underflow. Cannot dequeue from an empty queue.\n");
```

```
        return NULL;
    }
    *dequeuedValue = front->data;
    Node* temp = front;
    front = front->next;
    free(temp);
    return front;
}

// Function to display the elements of the linked list
void display(Node* head) {
    if (head == NULL) {
        printf("The list is empty.\n");
        return;
    }

    printf("List elements: ");
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}
```

Output:

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 3
Enter data to enqueue: 33

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 3
Enter data to enqueue: 44

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 3
Enter data to enqueue: 55

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 5
List elements: 33 44 55

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 4
Dequeued element: 33
```

```
Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 5
List elements: 33 44 55

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 4
Dequeued element: 33

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 33
Invalid choice. Please enter a valid option.

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice: 5
List elements: 44 55

Menu:
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
Enter your choice:
```