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

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

struct Node* head = NULL;

void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node
        ));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

int pop() {
    if (head == NULL) return -1;
    struct Node* temp = head;
    int val = temp->data;
    head = head->next;
    free(temp);
    return val;
}
```

```
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
1
Enter value:
12

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
1
Enter value:
13

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
```

```c
void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node
        ));
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
        return;
    }

    struct Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
}

int dequeue() {
    if (head == NULL) return -1;
    struct Node* temp = head;
    int val = temp->data;
    head = head->next;
    free(temp);
    return val;
```

```
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
2
Popped: 13

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
3
Enter value:
14

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
```

```
    free(temp);
    return val;
}

void display() {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, value;

    while (1) {
        printf("\n1. Push (Stack)\n2. Pop (Stack)\n3. Enqueue
            (Queue)\n4. Dequeue (Queue)\n5. Display\n6. Exit\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value:\n");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                value = pop();
                if (value == -1) printf("Stack Empty\n");
                else printf("Popped: %d\n", value);
                break;

            case 3:
                printf("Enter value:\n");
                scanf("%d", &value);
                enqueue(value);
                break;

            case 4:
                value = dequeue();
                if (value == -1) printf("Queue Empty\n");
                else printf("Dequeued: %d\n", value);
                break;

            case 5:
                display();
                break;

            case 6:
                return 0;
        }
    }
```

```
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
3
Enter value:
15

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
4
Dequeued: 12

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
5
14 -> 15 -> NULL
```

```
1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
4
Dequeued: 12

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
5
14 -> 15 -> NULL

1. Push (Stack)
2. Pop (Stack)
3. Enqueue (Queue)
4. Dequeue (Queue)
5. Display
6. Exit
```