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

typedef struct node
{
    int data;
    int priority;
    struct node *next;
} Node;

typedef struct
{
    Node *front;
} PQLL;

// Initialize an empty priority queue.
void initialize(PQLL *L)
{
    L->front = NULL;
}

// Insert an element with its priority into the priority queue.
void insert(PQLL *L, int data, int priority)
{
    Node *p = (Node *)malloc(sizeof(Node));
    if (p == NULL)
    {
        fprintf(stderr, "Memory allocation failed\n");
        exit(1);
    }
    p->data = data;
    p->priority = priority;
    p->next = NULL;

    if (L->front == NULL || priority < L->front->priority)
    {
        p->next = L->front;
        L->front = p;
    }
    else
    {
        Node *current = L->front;
        while (current->next != NULL && current->next->priority <= priority)
        {
            current = current->next;
        }
        p->next = current->next;
        current->next = p;
```

```
    }
}

// Remove and return the element with the highest priority from the priority
queue.
int deleteHighestPriority(PQLL *L)
{
    if (L->front == NULL)
    {
        printf("Priority queue is empty\n");
        exit(1);
    }

    Node *temp = L->front;
    int data = temp->data;
    L->front = L->front->next;
    free(temp);
    return data;
}

// Display the elements in the priority queue.
void display(PQLL L)
{
    Node *current = L.front;
    while (current != NULL)
    {
        printf("(%d, %d) ", current->data, current->priority);
        current = current->next;
    }
    printf("\n");
}

int main()
{
    PQLL l;
    initialize(&l);

    int choice;
    int data, priority;

    while (1)
    {
        printf("\nPriority Queue Menu:\n");
        printf("1. Insert an element\n");
        printf("2. Delete highest priority element\n");
        printf("3. Display priority queue\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
```

```
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                printf("Enter data and priority: ");
                scanf("%d %d", &data, &priority);
                insert(&l, data, priority);
                break;
            case 2:
                if (l.front == NULL)
                {
                    printf("Priority queue is empty\n");
                }
                else
                {
                    int highestPriority = deleteHighestPriority(&l);
                    printf("Deleted highest priority element: (%d, %d)\n",
highestPriority, 1);
                }
                break;
            case 3:
                printf("Priority Queue: ");
                display(l);
                break;
            case 4:
                printf("Exiting program\n");
                exit(0);
            default:
                printf("Invalid choice, please try again\n");
        }
    }

    return 0;
}
```

```
Priority Queue Menu:
1. Insert an element
2. Delete highest priority element
3. Display priority queue
4. Exit
Enter your choice: 1
Enter data and priority: 10 2

Priority Queue Menu:
1. Insert an element
2. Delete highest priority element
3. Display priority queue
4. Exit
Enter your choice: 1
Enter data and priority: 20 1

Priority Queue Menu:
1. Insert an element
2. Delete highest priority element
3. Display priority queue
4. Exit
Enter your choice: 1
Enter data and priority: 30 3
```

```
Priority Queue Menu:
1. Insert an element
2. Delete highest priority element
3. Display priority queue
4. Exit
Enter your choice: 2
Deleted highest priority element: (20, 1)

Priority Queue Menu:
1. Insert an element
2. Delete highest priority element
3. Display priority queue
4. Exit
Enter your choice: 3
Priority Queue: (10, 2) (30, 3)

Priority Queue Menu:
1. Insert an element
2. Delete highest priority element
3. Display priority queue
4. Exit
Enter your choice: 4
Exiting program
PS C:\Users\Mark Lopes>
```