priority-queue-using-linked-list.c

```
//implement priority queue using linked list.
#include <stdio.h>
#include <stdlib.h>

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

typedef struct Queue{
   node *front;
   node *rear;
}queue;

int rear(queue q){
   return q.rear->data;
}

int front(queue q){
   return q.front->data;
}

queue create(){
   queue q = {NULL,NULL};
   return q;
}

void enQueue(queue *q, int new_data, int priority){
   node* new_node = (node*)malloc(sizeof(node));
   new_node->data = new_data;
   new_node->priority = priority;
   if(q->rear == NULL){
      new_node->next = q->rear;
      q->front = new_node;
      q->rear = new_node;
      return;
   }
   node *temp = q->rear, *prev = NULL;
   while(temp != NULL && temp->priority < priority){
      prev = temp;
      temp = temp->next;
   }
   if(temp == NULL){
      new_node->next = NULL;
      q->front->next = new_node;
      q->front = new_node;
      return;
   }
   new_node->next = temp;
   if(prev == NULL){
      q->rear = new_node;
      return;
   }
   prev->next = new_node;
}
```

```
int deQueue(queue *q){
   if(q->rear == NULL){
      printf("Underflow.\n");
      return -1;
   }
   int data;
   node *temp = q->rear;
   if(q->rear == q->front){
      data = temp->data;
      *q = create();
      free(temp);
      return data;
   }
   while(temp->next != q->front){
      temp = temp->next;
   }
   temp->next = NULL;
   node *next = q->front;
   q->front = temp;
   data = next->data;
   free(next);
   return data;
}

void display(queue q){
   if(q.rear == NULL){
      printf("The Queue is Empty.");
   }
   node *temp = q.rear;
   while(temp != NULL){
      printf("%d(%d)",temp->data,temp->priority);
      temp = temp->next;
      if(temp != NULL){
         printf(" -> ");
      }
   }
   printf("\n");
}

void main(){
   int data, priority;
   int op;
   queue q = create();
   while(1){
      printf("1. enQueue, 2. deQueue, 3. Display, 4. Exit.\nEnter Your Choice: ");
      scanf("%d",&op);
      switch(op){
         case 1:
            printf("Enter the element data and priority.{data priority}: ");
            scanf("%d %d",&data, &priority);
            enQueue(&q, data, priority);
            break;
         case 2:
            printf("Removed %d\n",deQueue(&q));
            // deQueue(&q);
            break;
         case 3:
```

```
            display(q);
            break;
        case 4:
            printf("Oops..");
            return;
        default:
            printf("Wrong Input.\n");
    }
  }
}
```

OUTPUT

```
PS S:\WorkSpace\CollegeWork\DataStructure> gcc .\priority-queue-using-linked-list.c
PS S:\WorkSpace\CollegeWork\DataStructure> ./a
1. enQueue, 2. deQueue, 3. Display, 4. Exit.
Enter Your Choice: 1 12 4 1 15 1 1 17 7 1 18 2 1 19 9 1 19 3 1 41 2 1 65 4 1 100 8
1. enQueue, 2. deQueue, 3. Display, 4. Exit.
Enter Your Choice: 3
15(1) -> 41(2) -> 18(2) -> 19(3) -> 65(4) -> 12(4) -> 17(7) -> 100(8) -> 19(9)
1. enQueue, 2. deQueue, 3. Display, 4. Exit.
Enter Your Choice: 2
Removed 19
1. enQueue, 2. deQueue, 3. Display, 4. Exit.
Enter Your Choice: 2
Removed 100
1. enQueue, 2. deQueue, 3. Display, 4. Exit.
Enter Your Choice: 2
Removed 17
1. enQueue, 2. deQueue, 3. Display, 4. Exit.
Enter Your Choice: 4
Oops..
PS S:\WorkSpace\CollegeWork\DataStructure>
```