**DS LAB 8: Priority Queue Using Linked List**

Name : Vivian Vijay Ludrick

Branch: SE Comps A Batch C

Roll No: 9914

CODE:

#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*)

{

*// Create a new node*

Node\* p = (Node\*)malloc(sizeof(Node));

if (p == NULL) {

printf("Memory allocation failed\n");

exit(1);

}

p->data = *data*;

p->priority = *priority*;

p->next = NULL;

*// If the queue is empty or the new node has higher priority than the front node,*

*// insert the new node at the front*

if (*L*->front == NULL || *priority* < *L*->front->priority)

{

p->next = *L*->front;

*L*->front = p;

}

else

{

*// Find the appropriate position to insert the new node based on its priority*

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);

}

*// Remove the front node and return its data*

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, data, priority;

while (1) {

printf("Priority Queue Menu:\n");

printf("1. Insert an element\n");

printf("2. Delete the element with the highest priority\n");

printf("3. Display the priority queue\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the data and priority of the element to be inserted: ");

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 the program\n");

exit(0);

default:

printf("Invalid choice\n");

}

}

return 0;

}

OUTPUT:



