**/\*WAP to implement priority queue using linked list \*/**

**#include<iostream>**

**#include<cstdlib>**

**using namespace std;**

**struct node**

**{**

**int info,priority;**

**node \*next;**

**};**

**class PQueue**

**{**

**node \*Front,\*rear;**

**bool IsEmpty();**

**public:**

**PQueue():Front(NULL),rear(NULL) {}**

**void enqueue(int,int);**

**void dequeue();**

**void viewFront();**

**void displayPQueue();**

**};**

**bool PQueue::IsEmpty()**

**{**

**if(Front==NULL)**

**return true;**

**else**

**return false;**

**}**

**void PQueue::enqueue(int data,int pri)**

**{**

**node \*temp=new node;**

**if(temp==NULL)**

**cout<<"\n\nFailed to initialize the memory for new node.\n\n";**

**else**

**{**

**temp->info=data;**

**temp->priority=pri;**

**if(Front==NULL)**

**{**

**temp->next=Front;**

**Front=rear=temp;**

**}**

**else if(temp->priority<Front->priority)**

**{**

**temp->next=Front;**

**Front=temp;**

**}**

**else**

**{**

**node \*ptr;**

**ptr=Front;**

**while(ptr->next!=NULL && ptr->next->priority<=temp->priority)**

**ptr=ptr->next;**

**temp->next=ptr->next;**

**ptr->next=temp;**

**if(temp->next==NULL)**

**rear=temp;**

**}**

**}**

**}**

**void PQueue::dequeue()**

**{**

**if(IsEmpty())**

**cout<<"\n\nQueue underflow\n\n";**

**else**

**{**

**node \*temp;**

**temp=Front;**

**cout<<"\n\nThe dequeued element with priority is : \nElement = "<<Front->info<<"\tPriority = "<<Front->priority<<"\n\n";**

**if(Front==rear)**

**Front=rear=NULL;**

**else**

**Front=Front->next;**

**delete temp;**

**}**

**}**

**void PQueue::viewFront()**

**{**

**if(IsEmpty())**

**cout<<"\n\nQueue underflow\n\n";**

**else**

**cout<<"\n\nThe front element with priority is : \nElement = "<<Front->info<<"\tPriority = "<<Front->priority<<"\n\n";**

**}**

**void PQueue::displayPQueue()**

**{**

**if(IsEmpty())**

**cout<<"\n\nQueue underflow\n\n";**

**else**

**{**

**node \*temp;**

**temp=Front;**

**cout<<"\n\nElements of Priority Queue are : \nElement\t\tPriority\n";**

**while(temp!=NULL)**

**{**

**cout<<temp->info<<"\t\t"<<temp->priority<<endl;**

**temp=temp->next;**

**}**

**cout<<"\n\n";**

**}**

**}**

**int main()**

**{**

**int choice,num,priority;**

**PQueue q;**

**while(1)**

**{**

**cout<<"1. Enqueue\n2. Dequeue\n3. View front element\n4. View queue\n5. Exit\n\nEnter your choice : ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**{**

**while(1)**

**{**

**cout<<"\nEnter -1 to finish enqueue\nEnter the value: ";**

**cin>>num;**

**if(num==-1)**

**break;**

**cout<<"\nEnter priority for "<<num<<" : ";**

**cin>>priority;**

**q.enqueue(num,priority);**

**}**

**break;**

**}**

**case 2:**

**{**

**q.dequeue();**

**break;**

**}**

**case 3:**

**{**

**q.viewFront();**

**break;**

**}**

**case 4:**

**{**

**q.displayPQueue();**

**break;**

**}**

**default :**

**exit(0);**

**}**

**}**

**return 0;**

**}**

**/\*WAP to implement priority queue using linked list \*/**

**#include<iostream>**

**#include<cstdlib>**

**using namespace std;**

**struct node**

**{**

**int data;**

**int priority;**

**node\* next;**

**};**

**class prqueue**

**{**

**node \*head;**

**public:**

**prqueue()**

**{**

**head = NULL;**

**}**

**void enqueue(int n, int priority)**

**{**

**node \*newNode = new node;**

**newNode->data = n;**

**newNode->priority = priority;**

**if(head == NULL)**

**{**

**head = newNode;**

**head->next = NULL;**

**}**

**else**

**{**

**node \*ptr = head;**

**node \*preptr = NULL;**

**while(ptr->priority < priority)**

**{**

**preptr = ptr;**

**if(ptr->next == NULL)**

**{**

**break;**

**}**

**ptr = ptr->next;**

**}**

**if(preptr == NULL)**

**{**

**newNode->next = head;**

**head = newNode;**

**}**

**else if(priority <= ptr->priority)**

**{**

**preptr->next = newNode;**

**newNode->next = ptr;**

**}**

**else**

**{**

**newNode->next = ptr->next;**

**ptr->next = newNode;**

**}**

**}**

**}**

**void dequeue()**

**{**

**//Sankalpa\_Rijal's code**

**node \*ptr = head;**

**cout<<endl<<"The dequeued data is: "<<head->data<<endl;**

**head = head->next;**

**delete ptr;**

**}**

**void display\_prqueue()**

**{**

**if(head == NULL)**

**{**

**cout<<"\nThe list is empty!!"<<endl;**

**}**

**else**

**{**

**cout<<endl<<endl;**

**node \*ptr = head;**

**while(ptr != NULL)**

**{**

**cout<<" "<<ptr->data<<" ";**

**ptr = ptr->next;**

**}**

**cout<<endl<<endl;**

**}**

**}**

**};**

**int main()**

**{**

**prqueue queueobj;**

**int choose;**

**do**

**{**

**fflush(stdin);**

**cout<<"1. Enqueue."<<endl;**

**cout<<"2. Dequeue"<<endl;**

**cout<<"3. Exit"<<endl;**

**cout<<"\n\n\tChoose an option: ";**

**cin>>choose;**

**switch (choose)**

**{**

**case 1:**

**{**

**int val, priority;**

**char trash;**

**cout<<"\nEnter push val,priority: ";**

**cin>>val>>trash>>priority;**

**queueobj.enqueue(val,priority);**

**break;**

**}**

**case 2:**

**{**

**queueobj.dequeue();**

**break;**

**}**

**case 3:**

**{**

**exit(1);**

**break;**

**}**

**default :**

**{**

**cout<<"Invalid input";**

**break;**

**}**

**}**

**queueobj.display\_prqueue();**

**}**

**while (choose != 3);**

**return 0;**

**}**

**/\*WAP to implement priority queue using linked list \*/**

**#include<iostream>**

**using namespace std;**

**class Queue**

**{**

**struct node**

**{**

**int data;**

**int priority;**

**struct node \* next;**

**};**

**public:**

**struct node \* start;**

**struct node \* newnode,\* temp,\* ptr;**

**void creation()**

**{**

**newnode = new node;**

**cout<<"Enter the data for the queue(insert -1 to end the ): ";**

**cin>>newnode->data;**

**cout<<"Enter the priority of the data: ";**

**cin>>newnode->priority;**

**newnode->next=NULL;**

**if (start==NULL)**

**{**

**start=newnode;**

**temp=newnode;**

**}**

**else**

**{**

**temp->next=newnode;**

**temp=newnode;**

**}**

**do**

**{**

**enqueue();**

**}**

**while (newnode->data!=-1);**

**}**

**void enqueue()**

**{**

**newnode=new node;**

**cout<<"Enter the data to be stored in the queue: ";**

**cin>>newnode->data;**

**newnode->next=NULL;**

**if (newnode->data!=-1)**

**{**

**cout<<"Enter the priority of the data: ";**

**cin>>newnode->priority;**

**ptr=start;**

**if (newnode->priority<start->priority)**

**{**

**newnode->next=start;**

**start=newnode;**

**}**

**else**

**{**

**while(ptr->next!=NULL && ptr->next->priority<newnode->priority )**

**{**

**ptr=ptr->next;**

**}**

**newnode->next=ptr->next;**

**ptr->next=newnode;**

**}**

**}**

**}**

**void dequeue()**

**{**

**ptr=start->next;**

**delete start;**

**start=ptr;**

**}**

**void display\_queue()**

**{**

**ptr=start;**

**cout<<endl;**

**cout<<"-----------------------------------------------------------"<<endl;**

**cout<<"\n\nThe queue is: "<<endl;**

**cout<<"\t\t"<<ptr->data<<"|"<<ptr->priority;**

**while(ptr->next!=NULL)**

**{**

**ptr=ptr->next;**

**cout<<"\t"<<ptr->data<<"|"<<ptr->priority;**

**}**

**cout<<endl;**

**cout<<"-----------------------------------------------------------"<<endl;**

**}**

**};**

**int main()**

**{**

**class Queue q;**

**q.start=NULL;**

**int choice=0,c=0;**

**while(choice!=10)**

**{**

**c++;**

**cout<<"\n\nyour Choice please: "<<endl;**

**if (c==1)**

**{**

**cout<<"0-Creating a new queue "<<endl;**

**}**

**cout<<"1-Enqueue "<<endl;**

**cout<<"2-Dequeue "<<endl;**

**cout<<"10-Exit.\n"<<endl;**

**cout<<"\t\tyour choice: ";**

**cin>>choice;**

**switch (choice)**

**{**

**case 0:**

**q.creation();**

**break;**

**case 1:**

**q.enqueue();**

**break;**

**case 2:**

**q.dequeue();**

**break;**

**}**

**q.display\_queue();**

**}**

**cout<<"THANK YOU";**

**}**

**/\*WAP to implement priority queue using linked list \*/**

**#include<iostream>**

**using namespace std;**

**class linkList**

**{**

**struct Node**

**{**

**int data;**

**int priority;**

**Node \*next;**

**};**

**typedef struct Node\* nodeptr;**

**nodeptr head;**

**public:**

**linkList() //constructor**

**{**

**head=NULL;**

**}**

**void enqueue(int new\_data,int pi) //insert at the rear**

**{**

**nodeptr p,preptr;**

**nodeptr ptr=head;**

**preptr=NULL;**

**if(head==NULL)**

**{**

**nodeptr p;**

**p=new Node;**

**p->data= new\_data;**

**p->priority=pi;**

**p->next=head;**

**head=p;**

**}**

**else**

**{**

**while(ptr->priority<pi)**

**{**

**preptr=ptr;**

**if(ptr->next==NULL)**

**{**

**break;**

**}**

**ptr=ptr->next;**

**}**

**if(preptr==NULL)**

**{**

**p=new Node;**

**p->next=ptr;**

**p->data=new\_data;**

**p->priority=pi;**

**head=p;**

**}**

**else if(ptr->priority>=pi)**

**{**

**p=new Node;**

**preptr->next=p;**

**p->next=ptr;**

**p->data=new\_data;**

**p->priority=pi;**

**}**

**else**

**{**

**p=new Node;**

**ptr->next=p;**

**p->data =new\_data;**

**p->priority=pi;**

**p->next =NULL;**

**}**

**}**

**}**

**int dequeue() // delete from the front**

**{**

**nodeptr ptr=head;**

**if(head!=NULL)**

**{**

**head=ptr->next;**

**cout<<ptr->data<<" is Dequeued\n\n"<<endl;**

**delete ptr;**

**return ptr->data;**

**}**

**else**

**{**

**cout<<"Empty\n\n"<<endl;**

**return -1;**

**}**

**}**

**void display() // display the list**

**{**

**nodeptr p=head;**

**cout<<"\n\t=================X================"<<endl;**

**cout<<"\tadd"<<"\t\t\tprio"<<"\tdata"<<"\tnext"<<endl;**

**while(p!=NULL)**

**{**

**cout<<"\t"<<p<<"\t\t"<<p->priority<<"\t"<<p->data<<"\t"<<p->next<<endl;**

**p=p->next;**

**}**

**if(head==NULL)**

**{**

**cout<<"\tEmpty"<<endl;**

**}**

**cout<<"\tthats it"<<endl;**

**cout<<"\t=================X================\n"<<endl;**

**}**

**};**

**int main()**

**{**

**linkList li;**

**int x,p;**

**int choice=-1;**

**while(choice!=0)**

**{**

**cout<<"\n\nyour Choice please: "<<endl;**

**cout<<"1-Enqueue "<<endl;**

**cout<<"2-Dequeue "<<endl;**

**cout<<"0-Exit\n"<<endl;**

**cout<<"\t\tyour choice: ";**

**cin>>choice;**

**system("CLS");**

**cout<<"\tBEFORE LIST";**

**li.display();**

**switch (choice)**

**{**

**case 1:**

**cout<<"enter data to insert: ";**

**cin>>x;**

**cout<<"enter priority: ";**

**cin>>p;**

**li.enqueue(x,p);**

**break;**

**case 2:**

**li.dequeue();**

**break;**

**}**

**cout<<"\tAFTER LIST";**

**li.display();**

**}**

**cout<<"\n============X==========="<<endl;**

**cout<<"\t THANK YOU "<<endl;**

**return 0;**

**}**