\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data Structure Lab

CEN-391

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program 10

Code :-

#include <iostream>

#include <string.h>

using namespace std;

struct Priority\_Queue

{

    char process[4];

    int priority;

    Priority\_Queue \*next;

};

void isEmpty(int size)

{

    cout << "isEmpty...\n";

    if (size == 0)

        cout << "Empty" << endl;

    else

        cout << "Not Empty" << endl;

}

void Display(Priority\_Queue \*head, int size)

{

    cout << "Display...\n";

    if (size == 0)

    {

        cout << "Queue Is Empty" << endl;

        return;

    }

    while (head != nullptr)

    {

        cout << head->process << "(" << head->priority << ")"

             << "->";

        head = head->next;

    }

    cout << "Null\n";

    cout << endl;

}

void Enqueue(Priority\_Queue \*&head, Priority\_Queue \*&tail, int &size)

{

    cout << "Enqueue...\n";

    Priority\_Queue \*newnode = (Priority\_Queue \*)malloc(1 \* sizeof(Priority\_Queue));

    if (newnode == nullptr)

    {

        cout << "Memory Not Assigned" << endl;

        return;

    }

    size++;

    cout << "Enter The Priority : ";

    int priority;

    cin >> priority;

    fflush(stdin);

    cout << "Enter The Process Name : ";

    char process[4];

    gets(process);

    strcpy(newnode->process, process);

    newnode->priority = priority;

    newnode->next = nullptr;

    Priority\_Queue \*temp = head;

    if (head == nullptr)

    {

        head = newnode;

        tail = newnode;

    }

    else

    {

        if (temp->priority > newnode->priority)

        {

            newnode->next = head;

            head = newnode;

        }

        else if (tail->priority <= newnode->priority)

        {

            tail->next = newnode;

            tail = tail->next;

        }

        else

        {

            while (temp && temp->next)

            {

                if (temp->next->priority > newnode->priority)

                {

                    newnode->next = temp->next;

                    temp->next = newnode;

                    break;

                }

                temp = temp->next;

            }

        }

    }

    Display(head, size);

}

void Dequeue(Priority\_Queue \*&head, int &size)

{

    cout << "Dequeue...\n";

    if (size == 0)

    {

        cout << "Queue Underflow" << endl;

        return;

    }

    cout << head->process << "(" << head->priority << ")"

         << "\n";

    size--;

    Priority\_Queue \*todelete = head;

    head = head->next;

    delete todelete;

    Display(head, size);

}

void Front\_Rear(Priority\_Queue \*head, Priority\_Queue \*tail, int size)

{

    cout << "Front And Rear...\n";

    if (size == 0)

    {

        cout << "Queue Is Empty" << endl;

        return;

    }

    cout << "Front : " << head->process << endl;

    cout << "Rear : " << tail->process << endl;

}

void Total\_Element(int size)

{

    cout << "Total Elements In Priority Queue : " << size << endl;

}

void Bars()

{

    cout << "---------------------------------------------------------------\n";

}

bool Options(Priority\_Queue \*&head, Priority\_Queue \*&tail, int &size)

{

    int opt;

    cin >> opt;

    Bars();

    switch (opt)

    {

    case 1:

        Enqueue(head, tail, size);

        break;

    case 2:

        Dequeue(head, size);

        break;

    case 3:

        Front\_Rear(head, tail, size);

        break;

    case 4:

        isEmpty(size);

        break;

    case 5:

        Total\_Element(size);

        break;

    case 6:

        Display(head, size);

        break;

    case 7:

        cout << "Exit...\n";

        return 0;

    default:

        cout << "Invalid Input!\nTry Again!\n";

    }

    Bars();

    return 1;

}

void Menu()

{

    cout << "\_\_\_\_\_Operations\_On\_Priority\_Queue\_\_\_\_\_ \n";

    cout << "1.Enqueue \n";

    cout << "2.Dequeue \n";

    cout << "3.Front And Rear Element \n";

    cout << "4.isEmpty \n";

    cout << "5.Total Elements \n";

    cout << "6:Display \n";

    cout << "7.Exit \n";

    cout << "Enter Your Choice : ";

}

int main()

{

    system("cls");

    cout << "\_\_\_\_\_Vicky\_Gupta\_20BCS070\_\_\_\_\_\n\n";

    int size = 0;

    Priority\_Queue \*head = nullptr, \*tail = nullptr;

    while (true)

    {

        Menu();

        if (!Options(head, tail, size))

            break;

    }

    cout << "Exiting...\n";

    Bars();

    return 0;

}

Output :-

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated