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

Operating System Lab

CEN-493

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

Program - 1

Code :-

#include <iostream>

#include <string.h>

using namespace std;

struct Priority\_Queue

{

    char process\_name[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\_name << "|" << head->priority << "|"

             << "-->";

        head = head->next;

    }

    cout << "Null\n";

    cout << endl;

}

void Process\_Initialized(Priority\_Queue \*&new\_process)

{

    cout << "Enter The Priority : ";

    cin >> new\_process->priority;

    fflush(stdin);

    cout << "Enter The Process Name : ";

    gets(new\_process->process\_name);

    new\_process->next = nullptr;

}

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

{

    cout << "Insert Process...\n";

    Priority\_Queue \*new\_process = (Priority\_Queue \*)malloc(1 \* sizeof(Priority\_Queue));

    if (new\_process == nullptr)

    {

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

        return;

    }

    size++;

    Process\_Initialized(new\_process);

    Priority\_Queue \*temp = head;

    if (head == nullptr)

    {

        head = new\_process;

        tail = new\_process;

    }

    else

    {

        if (temp->priority > new\_process->priority)

        {

            new\_process->next = head;

            head = new\_process;

        }

        else if (tail->priority <= new\_process->priority)

        {

            tail->next = new\_process;

            tail = tail->next;

        }

        else

        {

            while (temp && temp->next)

            {

                if (temp->next->priority > new\_process->priority)

                {

                    new\_process->next = temp->next;

                    temp->next = new\_process;

                    break;

                }

                temp = temp->next;

            }

        }

    }

    Display(head, size);

}

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

{

    cout << "Execute\_Process...\n";

    if (size == 0)

    {

        cout << "Queue Underflow" << endl;

        return;

    }

    cout << "|" << head->process\_name << "|" << head->priority << "|"

         << "\n";

    size--;

    Priority\_Queue \*todelete = head;

    head = head->next;

    delete todelete;

    Display(head, size);

}

void Total\_Process(int size)

{

    cout << "Total No Of Process : " << 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:

        Insert\_Process(head, tail, size);

        break;

    case 2:

        Execute\_Process(head, size);

        break;

    case 3:

        Total\_Process(size);

        break;

    case 4:

        Display(head, size);

        break;

    case 5:

        cout << "Exit...\n";

        return 0;

    default:

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

    }

    Bars();

    return 1;

}

void Menu()

{

    cout << "\_\_\_\_\_Priority Scheduling Algorithm\_\_\_\_\_ \n";

    cout << "1.Insert Process \n";

    cout << "2.Execute \n";

    cout << "3.Total No Of Process \n";

    cout << "4.Display \n";

    cout << "5.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