Amina Qadeer

CE-42-A

359607

DS-LAB FINAL EXAM

CODE:

```
#include<iostream>
#include<string>
using namespace std;
struct DOCUMENT
{
    string Document_Status;
    int priority;
    DOCUMENT* NEXT;

};
class UserLinkedList
{
public:
    DOCUMENT* FRONT;
    int s;
    UserLinkedList()
    {
        FRONT = NULL;
        s = 0;
    }
                        //insertion
    void Insertion(DOCUMENT* doc)
    {
        int prior;
        string Document_Id;

        cout << "Please enter ID of document  " << endl;
        cin >> Document_Id;
        cout << "Please enter your priority for document  " << endl;
        cin >> prior;
        DOCUMENT* newnode = new DOCUMENT;
        newnode->Document_Status = Document_Id;
        newnode->priority = prior;

        if (FRONT == NULL || prior< FRONT->priority)
        {
```

```
            newnode->NEXT = FRONT;
            FRONT=newnode;
            s++;
        }
        else
        {
            DOCUMENT* Current_Node = FRONT;
            while (Current_Node->NEXT != NULL && Current_Node->NEXT->priority <=
prior)
            {
                Current_Node = Current_Node->NEXT;
            }
            newnode->NEXT = Current_Node->NEXT;
            Current_Node->NEXT = newnode;
            s++;
        }
    }
                    //deletion
    void Deletion()
    {
        DOCUMENT* deletenode = FRONT;
        if (IsEmpty())
        {
            cout << "THE DOCUMENT IS EMPTY!!" << endl;
        }
        else
        {
            FRONT = FRONT->NEXT;
            delete deletenode;
            deletenode = NULL;
            s--;
        }
    }
                  //remove the document
    void Remove(DOCUMENT* doc, string obj)
    {
        DOCUMENT* Current_Node =FRONT;
        if (status(doc))
        {
            doc->Document_Status =obj;
            while (Current_Node != NULL)
            {
                if (doc->Document_Status == Current_Node->NEXT->Document_Status)
                {
                    Current_Node->NEXT = Current_Node->NEXT->NEXT;
```

```
                    Current_Node = Current_Node->NEXT;
                    delete Current_Node;
                    Current_Node = NULL;
                    s--;

                }
                else
                {
                    Current_Node = Current_Node->NEXT;
                }
            }
        }
        else
        {
            cout << "This document  can not be deleted"<<endl;
        }

    }
           //CHECKING FOR THE STATUS OF SPOOLER
    bool status(DOCUMENT* OBJECT)
    {
        DOCUMENT* Current_Node = FRONT;
        string Document_Id;
        cout << "Please enter document ID to Check status " << endl;
        cin >> Document_Id;
        OBJECT->Document_Status = Document_Id;
        while (Current_Node != NULL)
        {
            if (Current_Node->Document_Status == OBJECT->Document_Status) {
                cout << "Document ID-NO " << OBJECT->Document_Status << " is
Spooling " << endl;
                return true;
            }
            else {
                Current_Node = Current_Node->NEXT;
            }
        }
        cout << "Entered ID_NO  for the current document is not  present  " <<
endl;
        return false;

    }
                               //IS EMPTY
    bool IsEmpty()
    {
```

```
        if (FRONT == NULL)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
};
struct User_Printer
{
    int user_id;
    User_Printer* linked;
    UserLinkedList N;
    User_Printer* NEXT;
};
class PrinterSpooler
{
public:
    DOCUMENT doc;
    User_Printer* head;
    int s;
    PrinterSpooler()
    {
        head = NULL;
        s = 0;
    }
    void insert(User_Printer* printer)
    {
        if (head == NULL)
        {
            head = printer;
        }
        else
        {
            User_Printer* newnode1 = head;
            while (newnode1->NEXT)
            {
                newnode1 = newnode1->NEXT;
            }
            newnode1->NEXT = printer;
        }
    }
    void Display(User_Printer* p)
```

```
{
    cout << "Please enter printer  ID as follow:" << endl;
    cout<<"1.LASER PRINTER"<<endl;
    cout<<"2.DOX PRINTER"<<endl;
    cin >> p->user_id;
    User_Printer* Current_Node = head;
    while (Current_Node != NULL)
    {
        if (Current_Node->user_id == p->user_id)
        {
            p->N.Insertion(&doc);
        }
        Current_Node = Current_Node->linked;

    }
    //Another node
    User_Printer* Newnode2 = new User_Printer;
    if (IsEmpty())
    {
        Newnode2->user_id = p->user_id;
        Newnode2->linked = NULL;
        head = Newnode2;
    }
    else
    {
        User_Printer* Current_Node1 = head;
        Newnode2->user_id = p->user_id;
        while (Current_Node1->linked != NULL)
        {
            Current_Node1 = Current_Node1->linked;

        }
        Newnode2->linked = NULL;
        Current_Node1->linked = Newnode2;
    }

}
void RemoveUser(int Document_Id)
{
    bool found = false;
    if (head->user_id == Document_Id)
    {
        found = true;
        User_Printer* Current_Node = head;
        head = Current_Node->linked;
```

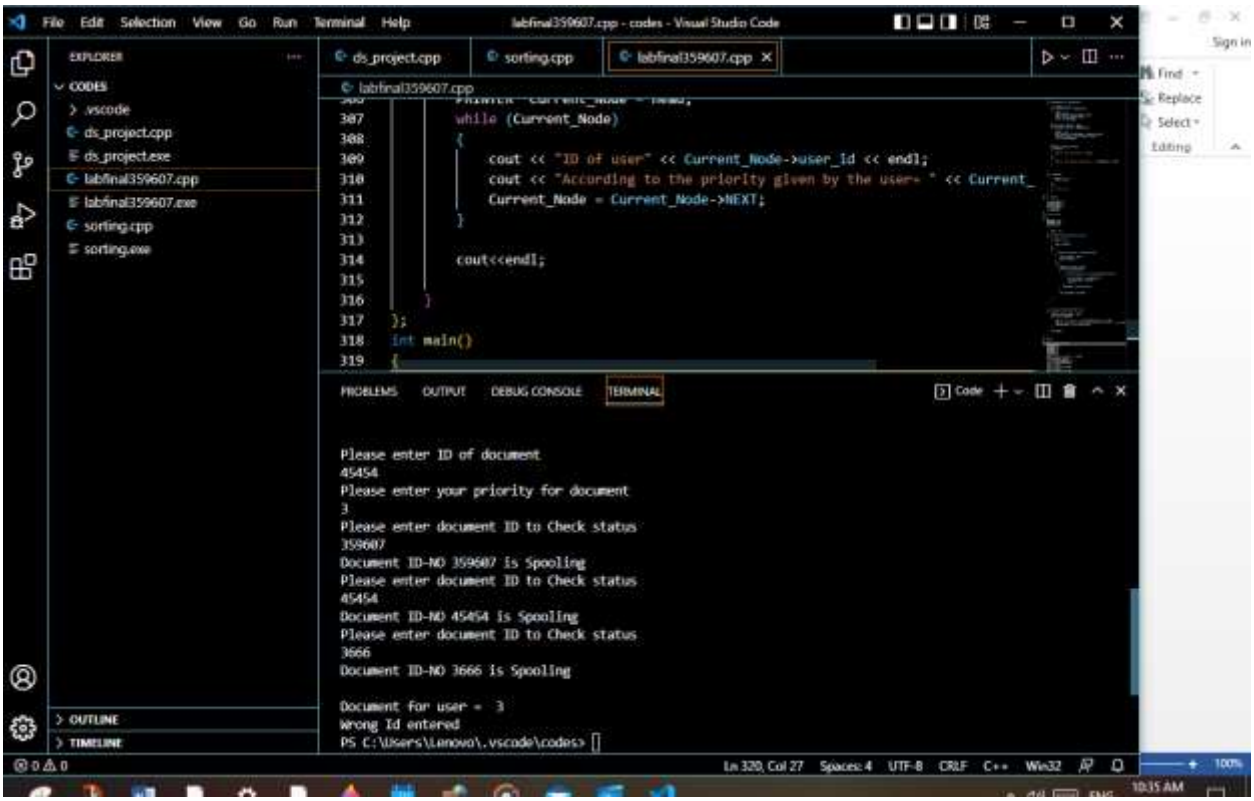
```
            delete Current_Node;

        }
        User_Printer* newnode = head;
        if (newnode->user_id == Document_Id)
        {
            found = true;
            User_Printer* Current_Node = newnode->linked;
            newnode->linked = Current_Node->linked;
            delete Current_Node;


        }
        newnode = newnode->linked;
        if (!found)
        {
            cout << "User not found!!" << endl;
        }
        else
        {
            cout << "User Removed successfully " << Document_Id << endl;
        }


    }

    bool IsEmpty()
    {
        if (head == NULL) {
            return true;
        }
        else
        {
            return false;
        }
    }
};
struct PRINTER
{
    int Displayer_no;
    PRINTER* NEXT;
    UserLinkedList* Y;
    int priority;
    int user_id;
};
```

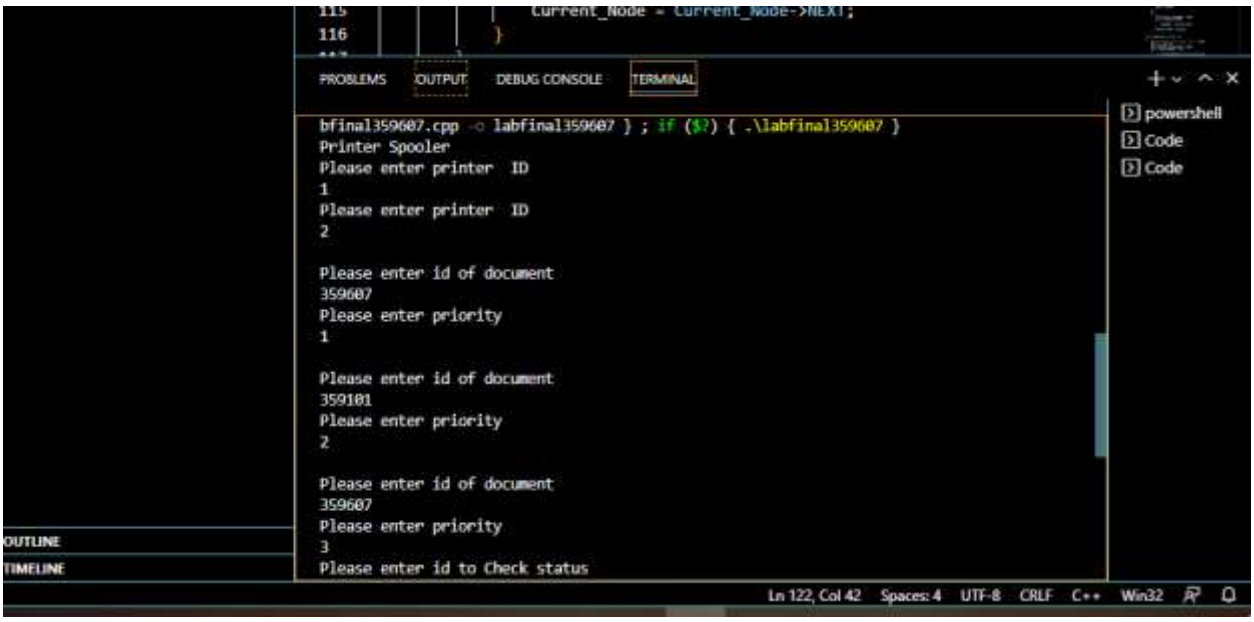
```
class PrinterList
{
public:
    PRINTER* head;
    DOCUMENT* doc;
    PrinterList()
    {

        head = NULL;
    }
    void Insert_Printer(PRINTER* person1)
    {
        if (head == NULL)
        {
            head = person1;
        }
        else
        {
            if (person1->priority > head->priority)
            {
                person1->NEXT = head;
                head = person1;
            }
            else
            {
                PRINTER* new_node = head;
                while (new_node->NEXT)
                {
                    if (person1->priority > new_node->NEXT->priority)
                    {
                        person1->NEXT = new_node->NEXT;
                        new_node->NEXT = person1;
                        return;
                    }
                    new_node = new_node->NEXT;
                }

                new_node->NEXT = person1;

            }
        }

    }
```

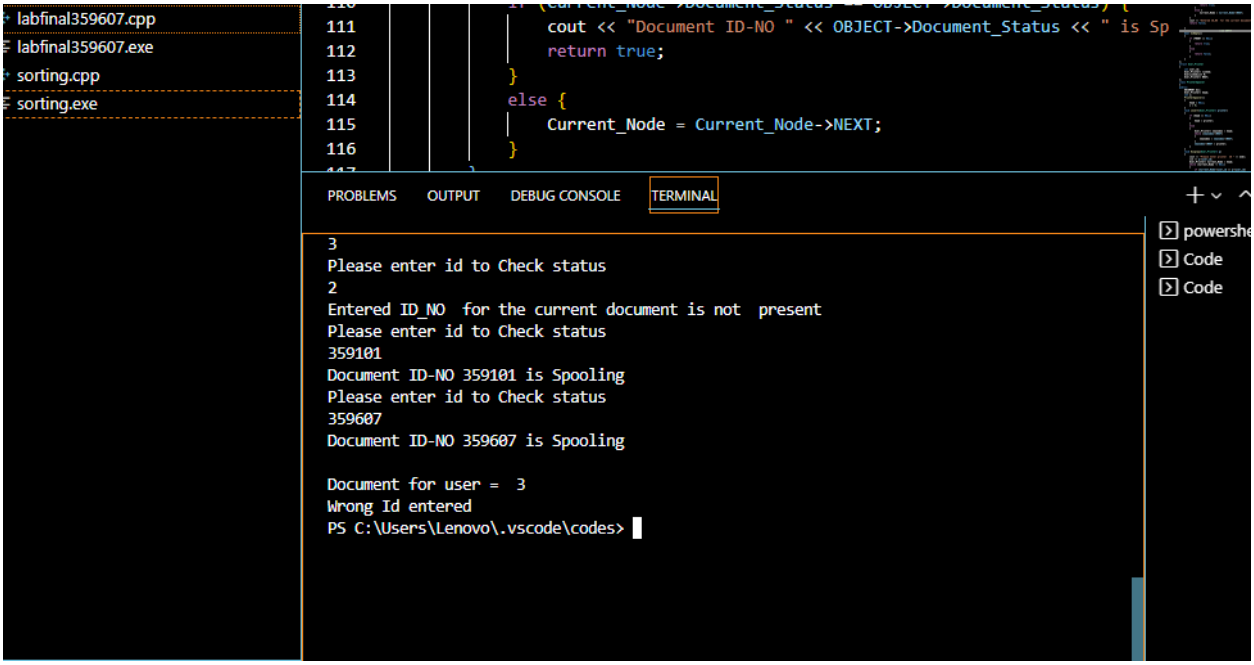
```
    void Display_Printer(DOCUMENT* doc)
    {
        PRINTER* Current_Node = head;
        while (Current_Node)
        {
            cout << "ID of user" << Current_Node->user_id << endl;
            cout << "According to the priority given by the user= " <<
Current_Node->priority << endl;
            Current_Node = Current_Node->NEXT;
        }

        cout<<endl;

    }
};
int main()
{
    User_Printer  Person1;
    PrinterList n;
    DOCUMENT L;
    DOCUMENT M;
    DOCUMENT N;
    cout<<endl;
    cout << "Printer Spooler" << endl;
    PrinterSpooler print;
    print.Display(&Person1);
    print.Display(&Person1);
    cout << endl;
    Person1.N.Insertion(&M);
    cout << endl;
    Person1.N.Insertion(&N);
    cout << endl;
    Person1.N.Insertion(&L);
    Person1.N.status(&N);
    Person1.N.status(&L);
    Person1.N.status(&M);
    cout << endl;
    cout << "Document for user =  " << Person1.N.s << endl;
    if (Person1.N.IsEmpty())
    {
        print.RemoveUser(80);
    }
    else
    {
        cout << "Wrong Id entered" << endl;
```

```
    }
    return 0;
}
```

Output:




Current_Node = Current_Node->NEXT;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
bfinal359607.cpp -o labfinal359607 } ; if ($?) { .\labfinal359607 }
Printer Spooler
Please enter printer ID
1
Please enter printer ID
2
Please enter id of document
359607
Please enter priority
1
Please enter id of document
359101
Please enter priority
2
Please enter id of document
359607
Please enter priority
3
Please enter id to Check status
powershell
Code
Code
OUTLINE
TIMELINE
Ln 122, Col 42
Spaces: 4
UTF-8
CRLF
C++
Win32


labfinal359607.cpp
labfinal359607.exe
sorting.cpp
sorting.exe
cout << "Document ID-NO " << OBJECT->Document_Status << " is Sp
return true;
}
else {
Current_Node = Current_Node->NEXT;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
3
Please enter id to Check status
2
Entered ID_NO for the current document is not present
Please enter id to Check status
359101
Document ID-NO 359101 is Spooling
Please enter id to Check status
359607
Document ID-NO 359607 is Spooling
Document for user = 3
Wrong Id entered
PS C:\Users\Lenovo\.vscode\codes>
Code
Code