DATA STRUCTURES LABORATORY

Course Code: 1BCSL306

VTU 2025 Scheme – Semester 3

PART – B

Open-Ended Experiment

EXPERIMENT – 08

Student Record Management

Using Singly Linked List

Simple Code • Easy to Remember • Exam Friendly

Experiment 08

1. Aim

Aim

To develop a menu-driven C program using a Singly Linked List to manage university student records containing USN, Name, Semester and Department.

2. VTU Prescribed Experiment

VTU Prescribed Experiment

A University wants to manage student records effectively.
Each student record contains:

- USN
- Name
- Semester (1–8)
- Department

Choose suitable data structure (Singly Linked List / Doubly Linked List / Circular Linked Lists) and develop a menu-driven C program to manage these records with the following operations:

a. Create a list of student records (choose any insertion method: beginning / end).

b. View all student records.

c. Display student records for a given semester.

d. Insert the new student in the list after the given student's name.

e. Delete a student record based on USN or Name.

Note:
For CIE Test and SEE, any ONE option from (c) to (e) may be asked along with options (a) and (b).

3. Simple Theory

A **Singly Linked List** is a collection of nodes.

Each node contains:

$$\boxed{Student\ Data + Next\ Pointer}$$

For this experiment each node stores:

$$\boxed{USN + Name + Semester + Department + Next}$$

4. Student Node Structure

```
struct Student
{
    char usn[20];
    char name[30];
    int sem;
    char dept[20];

    struct Student *next;
};
```

Each node points to the next student record.

Example:

$$\boxed{S1 \rightarrow S2 \rightarrow S3 \rightarrow NULL}$$

5. Why Singly Linked List?

Singly Linked List is suitable because:

- Student records can be added dynamically.
- Memory is allocated only when required.
- Insertion is easy.
- Deletion is easy.
- Records can be traversed sequentially.

6. Algorithm

6.1. Create Student List

1. Start.
2. Read number of students.
3. Create a new node for each student.

4. Read USN, Name, Semester and Department.
5. Insert each node at the end of the list.
6. Repeat until all records are inserted.

6.2. View All Student Records

1. Start from head.
2. Display student details.
3. Move to next node.
4. Repeat until NULL is reached.

6.3. Display Students of Given Semester

1. Read semester number.
2. Start from head.
3. Compare:

$$\boxed{student.sem == given\ semester}$$

4. If equal, display the student record.
5. Continue until end of list.

6.4. Insert After Given Student Name

1. Read student name to search.
2. Traverse the list.
3. If the name is found, create a new node.
4. Insert the new node after the matched student.

6.5. Delete Student by USN

1. Read USN to delete.
2. Search the list.
3. Keep previous and current pointers.
4. If record is found, change links.
5. Delete the node using `free()`.

Easy Exam Memory

Remember the complete experiment as:

$$\boxed{Create \rightarrow Display \rightarrow Semester \rightarrow Insert \rightarrow Delete}$$

Main node fields:

$$\boxed{USN,\ Name,\ Semester,\ Department,\ Next}$$

7. C Program – Easy Student Version

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct Student
{
    char usn[20];
    char name[30];
    int sem;
    char dept[20];
    struct Student *next;
};

struct Student *head=NULL;

struct Student* newStudent()
{
    struct Student *p;

    p=(struct Student*)malloc(sizeof(struct Student));

    printf("Enter USN: ");
    scanf("%s",p->usn);

    printf("Enter Name: ");
    scanf("%s",p->name);

    printf("Enter Semester: ");
    scanf("%d",&p->sem);

    printf("Enter Department: ");
    scanf("%s",p->dept);

    p->next=NULL;

    return p;
}

void create()
{
    struct Student *p,*temp;
    int n,i;

    printf("Enter number of students: ");
    scanf("%d",&n);

    for(i=0;i<n;i++)
    {
        printf("\nEnter Student %d Details\n",i+1);

```

```
        p=newStudent();

        if(head==NULL)
        {
            head=p;
        }
        else
        {
            temp=head;

            while(temp->next!=NULL)
                temp=temp->next;

            temp->next=p;
        }
    }
}

void display()
{
    struct Student *temp=head;

    if(head==NULL)
    {
        printf("List is Empty\n");
        return;
    }

    while(temp!=NULL)
    {
        printf("\nUSN: %s",temp->usn);
        printf("\nName: %s",temp->name);
        printf("\nSemester: %d",temp->sem);
        printf("\nDepartment: %s\n",temp->dept);

        temp=temp->next;
    }
}

void displaySem()
{
    struct Student *temp=head;
    int sem,found=0;

    printf("Enter Semester: ");
    scanf("%d",&sem);

    while(temp!=NULL)
    {
        if(temp->sem==sem)
        {
            printf("\n%s  %s  %d  %s\n",
            temp->usn,temp->name,
```

```
            temp->sem,temp->dept);

            found=1;
        }

        temp=temp->next;
    }

    if(found==0)
        printf("No Student Found\n");
}

void insertAfter()
{
    struct Student *temp=head,*p;
    char name[30];

    printf("Enter Student Name: ");
    scanf("%s",name);

    while(temp!=NULL &&
          strcmp(temp->name,name)!=0)
    {
        temp=temp->next;
    }

    if(temp==NULL)
    {
        printf("Student Not Found\n");
        return;
    }

    printf("Enter New Student Details\n");

    p=newStudent();

    p->next=temp->next;
    temp->next=p;

    printf("Student Inserted\n");
}

void deleteStudent()
{
    struct Student *temp=head,*prev=NULL;
    char usn[20];

    printf("Enter USN to Delete: ");
    scanf("%s",usn);

    while(temp!=NULL &&
          strcmp(temp->usn,usn)!=0)
    {
```

```
        prev=temp;
        temp=temp->next;
    }

    if(temp==NULL)
    {
        printf("Student Not Found\n");
        return;
    }

    if(prev==NULL)
        head=temp->next;
    else
        prev->next=temp->next;

    free(temp);

    printf("Student Deleted\n");
}

int main()
{
    int ch;

    do
    {
        printf("\n1.Create Student List");
        printf("\n2.View All Students");
        printf("\n3.Display by Semester");
        printf("\n4.Insert After Name");
        printf("\n5.Delete by USN");
        printf("\n6.Exit");

        printf("\nEnter Choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                create();
                break;

            case 2:
                display();
                break;

            case 3:
                displaySem();
                break;

            case 4:
                insertAfter();
                break;
```

```
            case 5:
                deleteStudent();
                break;

            case 6:
                printf("Program Ended\n");
                break;

            default:
                printf("Invalid Choice\n");
        }

    }while(ch!=6);

    return 0;
}
```


8. Simple Program Explanation

8.1. newStudent()

This function:

- Allocates memory.
- Reads USN.
- Reads Name.
- Reads Semester.
- Reads Department.

Finally:

```
p->next=NULL;
```

8.2. create()

Records are inserted at the end.

If:

```
head==NULL
```

the first student becomes head.

Otherwise traversal continues until:

```
temp->next==NULL
```

and the new node is attached.

8.3. display()

Traversal starts with:

```
temp=head;
```

and continues while:

```
temp!=NULL
```

8.4. displaySem()

Important condition:

```
if(temp->sem==sem)
```

Only students belonging to the entered semester are displayed.

8.5. insertAfter()

The name is compared using:

```
strcmp(temp->name,name)
```

After finding the student:

```
p->next=temp->next;
temp->next=p;
```

These two lines perform insertion.

8.6. deleteStudent()

Two pointers are used:

$$\boxed{prev}$$

and:

$$\boxed{temp}$$

If the first node is deleted:

```
head=temp->next;
```

Otherwise:

```
prev->next=temp->next;
```

Finally:

```
free(temp);
```

Easy Exam Memory

Create:

$$\boxed{New\ Node \rightarrow Traverse\ End \rightarrow Insert}$$

Display:

$$\boxed{Head \rightarrow Next \rightarrow NULL}$$

Semester:

$$\boxed{temp->sem == sem}$$

Insert after Name:

$$\boxed{p->next = temp->next}$$

$$\boxed{temp->next = p}$$

Delete:

$$\boxed{prev->next = temp->next}$$

9. Sample Input and Output

```

1.Create Student List
2.View All Students
3.Display by Semester
4.Insert After Name
5.Delete by USN
6.Exit

Enter Choice: 1

Enter number of students: 2

Enter Student 1 Details
Enter USN: 1SC25CS001
Enter Name: Ravi
Enter Semester: 3
Enter Department: CSE

Enter Student 2 Details
Enter USN: 1SC25CS002
Enter Name: Anu
Enter Semester: 5
Enter Department: CSE


Enter Choice: 2

USN: 1SC25CS001
Name: Ravi
Semester: 3
Department: CSE

USN: 1SC25CS002
Name: Anu
Semester: 5
Department: CSE
```


10. Display by Semester Example

Input:

```

Enter Choice: 3
Enter Semester: 3
```

Output:

```

1SC25CS001  Ravi  3  CSE
```

11. Insert After Name Example

Suppose the list is:

$$\boxed{Ravi \rightarrow Anu \rightarrow NULL}$$

Insert Kiran after Ravi.

Then:

$$\boxed{Ravi \rightarrow Kiran \rightarrow Anu \rightarrow NULL}$$

12. Delete Example

Suppose:

$$\boxed{Ravi \rightarrow Kiran \rightarrow Anu}$$

Delete Kiran.

After deletion:

$$\boxed{Ravi \rightarrow Anu \rightarrow NULL}$$

13. Simple Test Cases

Test	Operation	Expected Result
1	Create 3 records	Three student nodes are created.
2	View All	All records are displayed.
3	Search Semester 3	Only Semester 3 students are displayed.
4	Insert after Ravi	New student appears after Ravi.
5	Delete existing USN	Student node is removed.
6	Delete invalid USN	Student Not Found is displayed.

14. Result

Result

Thus, a menu-driven C program for managing University Student Records using a Singly Linked List was successfully implemented.
The program performs:

- Create student records.
- View all student records.
- Display students of a given semester.
- Insert a student after a given student name.
- Delete a student record using USN.

15. Important Viva Questions

Important Viva Questions

1. What is a linked list?
A linked list is a collection of nodes connected using pointers.

2. Which linked list is used in this program?
Singly Linked List.

3. What information is stored in each node?
USN, Name, Semester, Department and Next pointer.

4. What is head?
Head is a pointer to the first node of the list.

5. What does NULL indicate?
It indicates the end of the linked list.

6. Which function allocates memory dynamically?

`malloc()`

7. Which function releases memory?

`free()`

8. Which function compares strings?

`strcmp()`

9. Which header file contains strcmp()?

`string.h`

10. How do we traverse a linked list?
Start at head and repeatedly follow the next pointer until NULL.

11. How is semester filtering performed?
By comparing:

$$temp->sem == sem$$

12. How do we insert after a node?
Using:

$$p->next = temp->next$$

and:

$$temp->next = p$$

13. Why is previous pointer required during deletion?
It is used to connect the node before the deleted node to the next node.

14. What is the advantage of linked lists over arrays?
Linked lists can grow dynamically.

16. 1-Minute Exam Revision

Easy Exam Memory

Node Fields

$$\boxed{USN + Name + Semester + Department + Next}$$

Empty List

$$\boxed{head = NULL}$$

Traversal

$$\boxed{while(temp! = NULL)}$$

Semester Search

$$\boxed{temp- > sem == sem}$$

Insert After

$$\boxed{p- > next = temp- > next}$$

$$\boxed{temp- > next = p}$$

Delete

$$\boxed{prev- > next = temp- > next}$$

17. Easy Code Memory Map

Remember

Remember only six things:

$$\boxed{Student\ Structure}$$

$$\boxed{newStudent()}$$

$$\boxed{create()}$$

$$\boxed{display()}$$

$$\boxed{displaySem()}$$

$$\boxed{insertAfter() \rightarrow deleteStudent()}$$

Complete flow:

$$\boxed{Create \rightarrow View \rightarrow Search \rightarrow Insert \rightarrow Delete}$$

SEARCH CREATORS

1BCSL306 – Data Structures Laboratory

PART – B

EXPERIMENT 08
COMPLETED

Student Record Management

Using Singly Linked List

Create → Display → Search → Insert → Delete

Your VTU Study Partner