

# BÀI 4

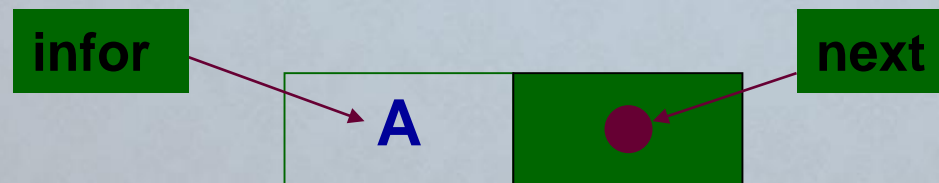
## DANH SÁCH TUYỂN TÍNH

### 4.4. DANH SÁCH NỐI ĐƠN



## 4.4.1. Khái niệm danh sách móc nối đơn

- Nguyên tắc tạo thành danh sách
  - Danh sách được tạo thành từ các phần tử gọi là nút (*Node*)
  - Các **node** có thể nằm bất kỳ đâu trong bộ nhớ
  - Mỗi **node** là một cấu trúc gồm 2 thành phần:
    - **infor** chứa thông tin của 1 phần tử của danh sách L
    - **next** là một con trỏ, nó trỏ vào node đứng sau



Một **node** trong danh sách

## Khái niệm danh sách móc nối đơn (tt)

- Ví dụ



Một **node** trong danh sách sinh viên

1089 là địa chỉ vùng nhớ của node đứng sau



## Khái niệm danh sách móc nối đơn (tt)

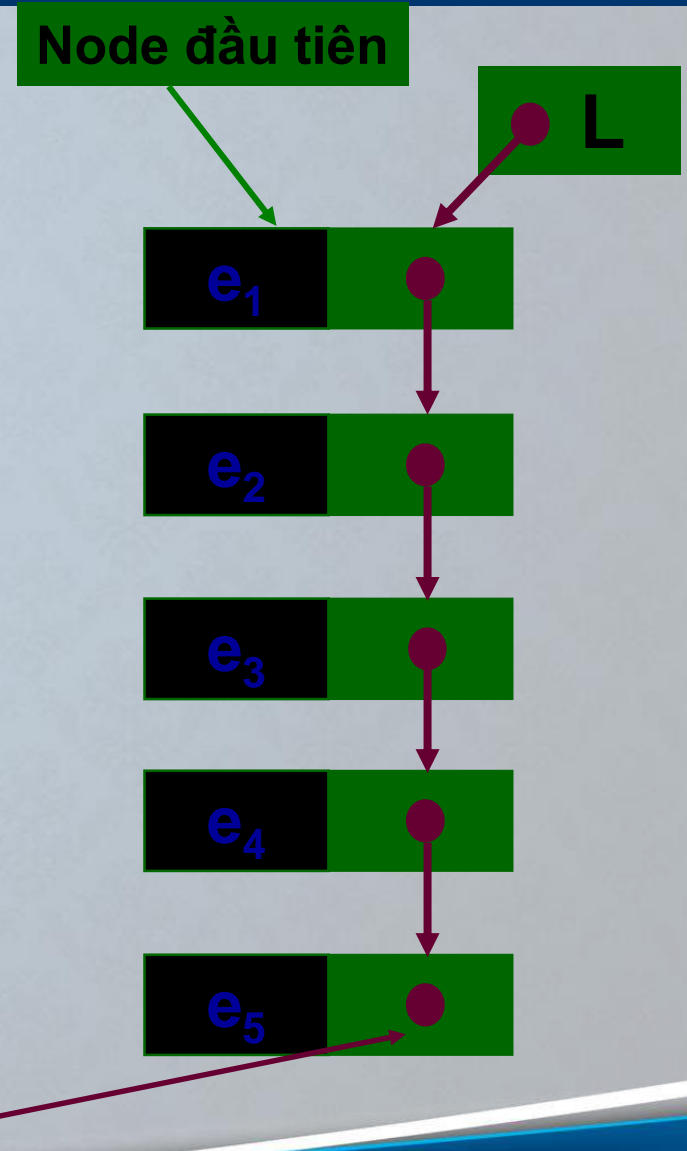
- Danh sách  $L = \{e_1, e_2, e_3, e_4, e_5\}$  được lưu trữ dưới dạng móc nối đơn
- Để truy nhập vào các node trong danh sách ta phải đi từ node đầu tiên
- Cần một con trỏ, trỏ vào node đầu của danh sách
- Phần tử cuối cùng của danh sách có next = NULL

L trỏ vào node đầu tiên của danh sách khi đó  
Để truy xuất vào thông tin của phần tử ta viết

**L->infor**

Để chỉ ra phần tử đứng sau ta viết

**L->next**



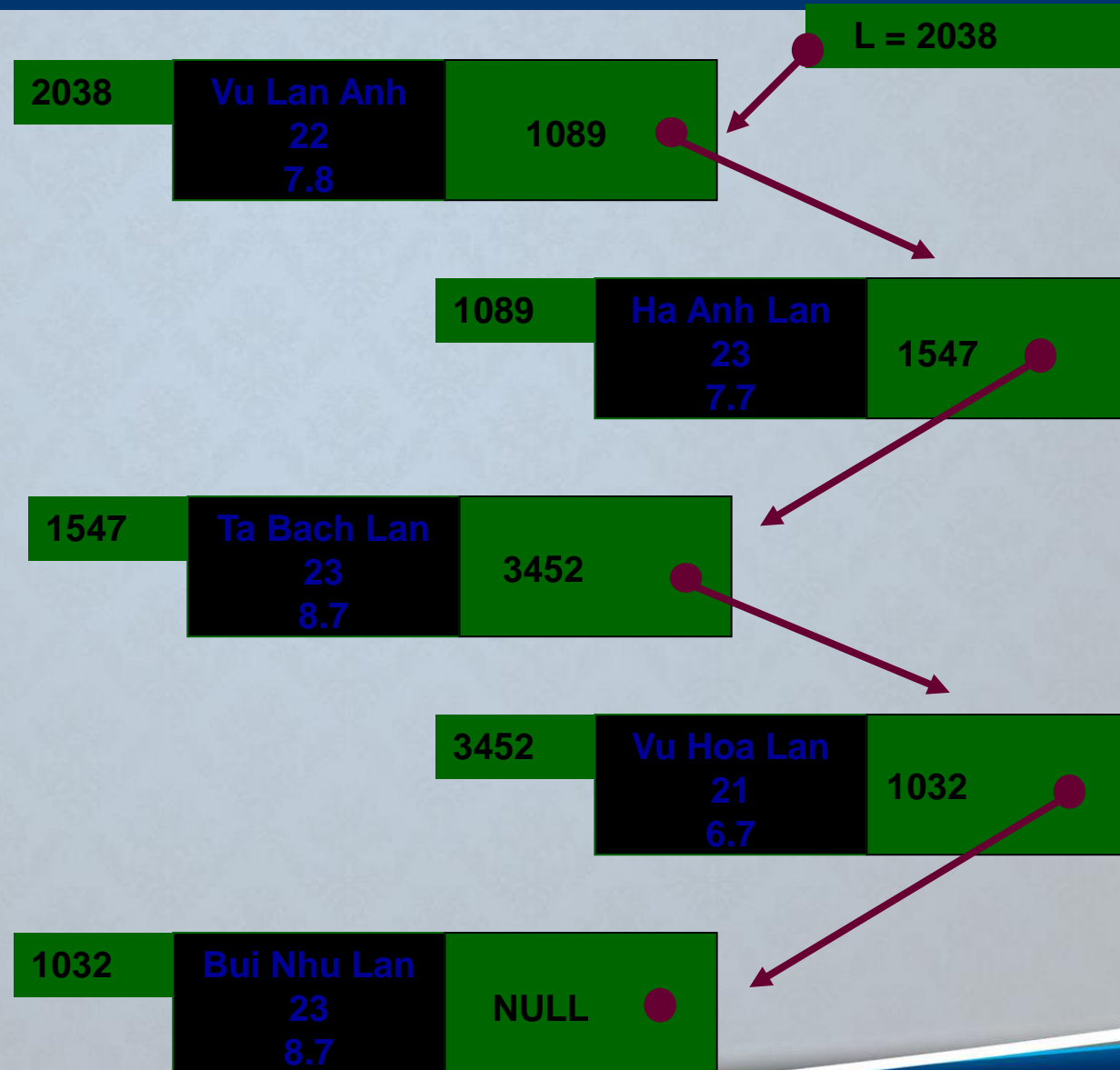
## Khái niệm danh sách móc nối đơn (tt) – Ví dụ

- Cho danh sách sinh viên

STT	Họ và tên	Tuổi	Điểm TK
1	Vũ Lan Anh	22	7.8
2	Hà Anh Lan	23	7.7
3	Tạ Bạch Lan	23	8.7
4	Vũ Hoa Lan	21	6.7
5	Bùi Như Lan	23	8.7

- Danh sách được lưu trữ trong bộ nhớ máy tính dưới dạng danh sách móc nối đơn.

## Ví dụ



## 4.4.2. Ưu và nhược điểm của danh sách nối đơn

- **Ưu điểm:**
  - Tiết kiệm bộ nhớ.
  - Các thao tác thêm và xóa thực hiện nhanh vì không phải dịch chuyển các phần tử.
- **Nhược điểm:**
  - Việc truy xuất vào các phần tử chậm vì luôn phải xuất phát từ phần tử đầu tiên.
  - Chỉ duyệt được danh sách theo một chiều nhất định, từ trên xuống.
  - Các thao tác khá phức tạp, khó hiểu với người mới lập trình.

## 4.4.3. Khai báo cấu trúc dữ liệu

Khai báo kiểu dữ liệu phần tử

```
struct DataType{  
    //Dữ liệu phần tử;  
};
```

Khai báo kiểu con trỏ Node

```
typedef Node *TRO;
```

Con trỏ L trỏ vào Node đầu

```
TRO L;
```

Khai báo kiểu dữ liệu Node

```
struct Node{  
    DataType infor;  
    Node *next;  
};
```

L = NULL -> ds L rỗng



## Khai báo cấu trúc dữ liệu (tt) – Ví dụ

### Khai báo kiểu dữ liệu phần tử

```
struct SinhVien{  
    int id;  
    char hoTen[30];  
    int tuoi;  
    float diemTk;  
};
```

### Khai báo kiểu dữ liệu Node

```
struct Node{  
    SinhVien infor;  
    Node *next;  
};
```

### Khai báo kiểu con trỏ Node

```
typedef Node *TRO;
```

### Con trỏ L trỏ vào Node đầu

```
TRO L;
```

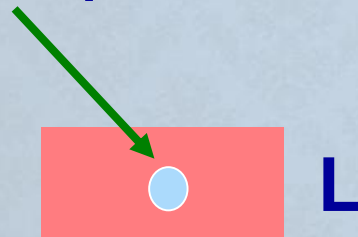


## 4.4.4. Các phép toán trên danh sách

- Khởi tạo danh sách rỗng
- Kiểm tra danh sách rỗng
- Duyệt danh sách
- Tìm kiếm một node trên danh sách
- Bổ sung node mới vào đầu danh sách
- Bổ sung node mới vào sau một node
- Xóa node đầu danh sách
- Xóa node đứng sau một node trong danh sách
- Sắp xếp danh sách

## 4.4.4.1. Khởi tạo danh sách rỗng

Giá trị **NULL**

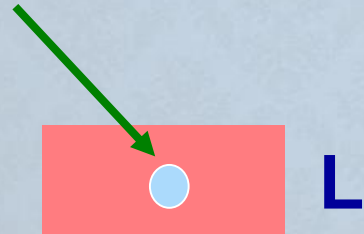


Danh sách nối đơn rỗng

```
void creat(TRO &L){  
    L = NULL;  
}
```

## 4.4.4.2. Kiểm tra danh sách rỗng

Giá trị **NULL**



Danh sách nối đơn rỗng

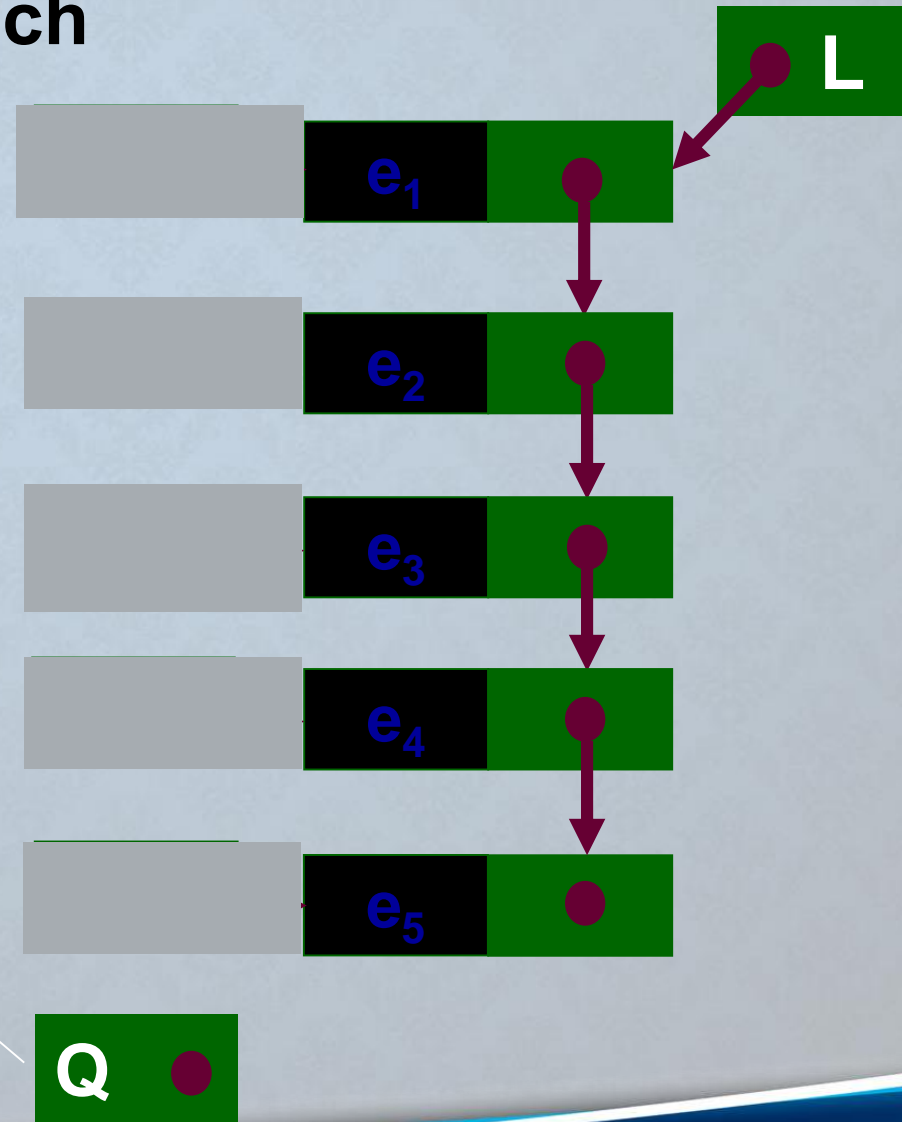
```
int empty(TRO L){  
    return L == NULL;  
}
```



### 4.4.4.3. Duyệt danh sách

1. Nếu danh sách không rỗng, cho con trỏ Q trở vào node đầu tiên:  $Q = L$ ;
2. Nếu  $Q \neq \text{NULL}$  thì (thực hiện yêu cầu) và chuyển Q xuống node ngay sau nó:  
 $Q = Q \rightarrow \text{next}$ ;
3. Lặp lại bước 2

**Q = NULL**



## Duyệt danh sách (tt)

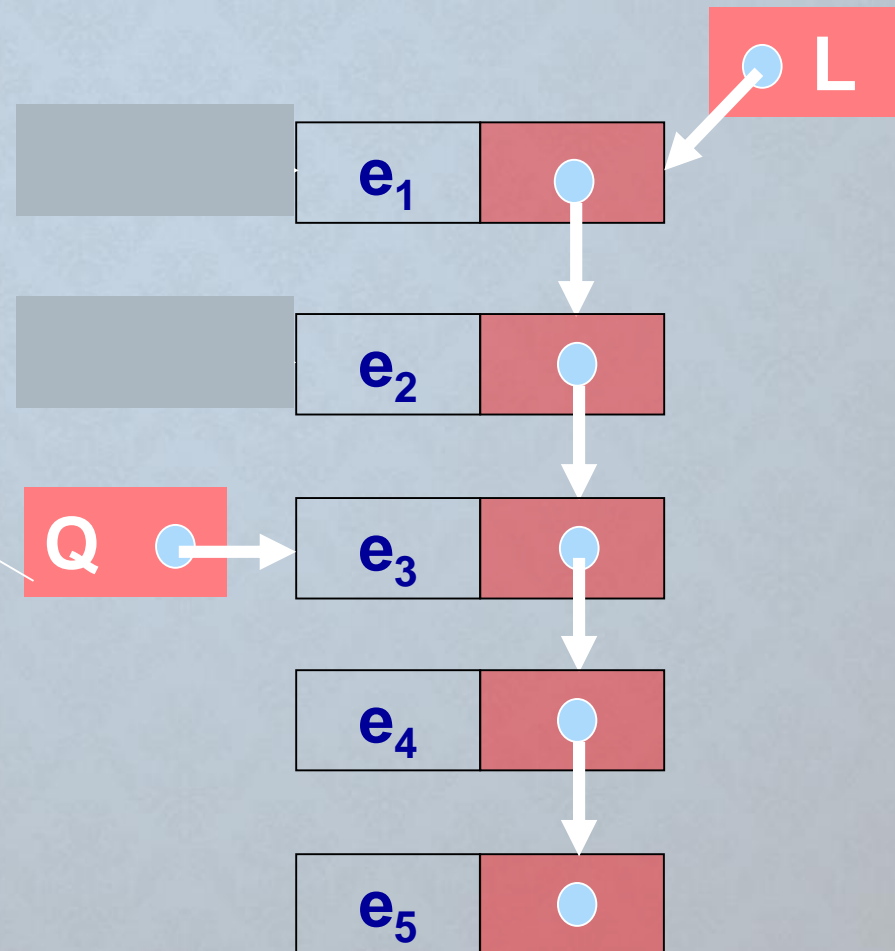
- Hàm duyệt danh sách như sau

```
void travel(TRO L) {  
    TRO Q;  
    if (!empty(L)){  
        Q = L;  
        while (Q != NULL){  
            //Statement  
            Q = Q->next;  
        }  
    }  
}
```

#### 4.4.4.4. Tìm kiếm một nút trên danh sách

Giả sử cần tìm node có **infor** là **e3** trong danh sách.

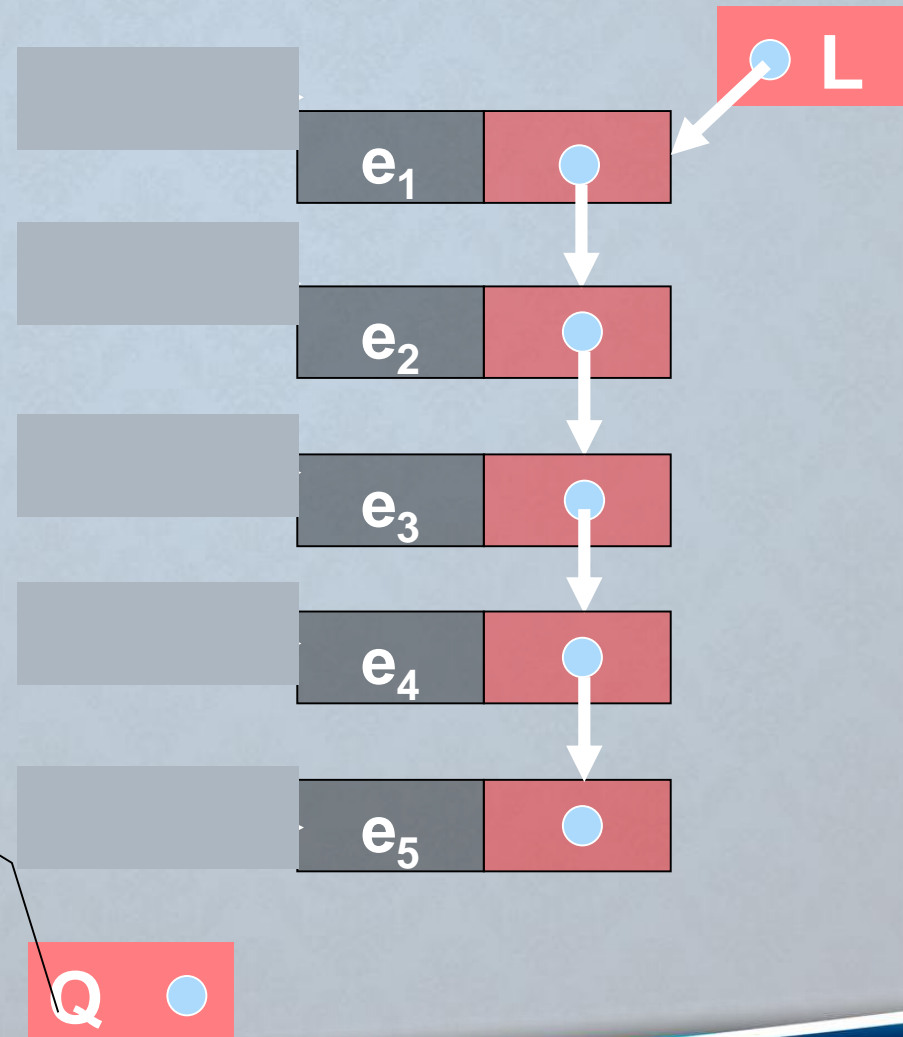
Tìm thấy và con trỏ **Q** trở vào node tìm được



## Tìm kiếm một nút trên danh sách (tt)

3. Giả sử cần tìm node có **infor** là **e7** trong danh sách.

**Không tìm thấy  
Q = NULL**





## Tìm kiếm một nút trên danh sách (tt)

1. Nếu danh sách không rỗng, cho con trỏ Q trở vào node đầu tiên:  $Q = L$ ;
2. Nếu  $(Q \neq \text{NULL})$  và (chưa trở vào node cần tìm) thì (có thể thực hiện yêu cầu) và chuyển Q xuống node ngay sau nó:  
 $Q = Q \rightarrow \text{next}$ ;
3. Lặp lại bước 2
4. Trả về con trỏ Q: **return** Q;

## Tìm kiếm một nút trên danh sách (tt)

```
TRO search(TRO L) {  
    TRO Q = L;  
    while (Q != NULL && (ĐKTK chưa thỏa))  
        Q = Q->next;  
    return Q;  
}
```

Hàm search trả về NULL nếu không tìm thấy, ngược lại trả về con trỏ trỏ vào node tìm được

#### 4.4.4.5. Chèn một nút vào đầu danh sách

Danh sách có phần tử đầu tiên được trỏ bởi con trỏ L

Giả sử dữ liệu của phần tử lưu trong biến **elem**

**elem**

**e<sub>1</sub>**

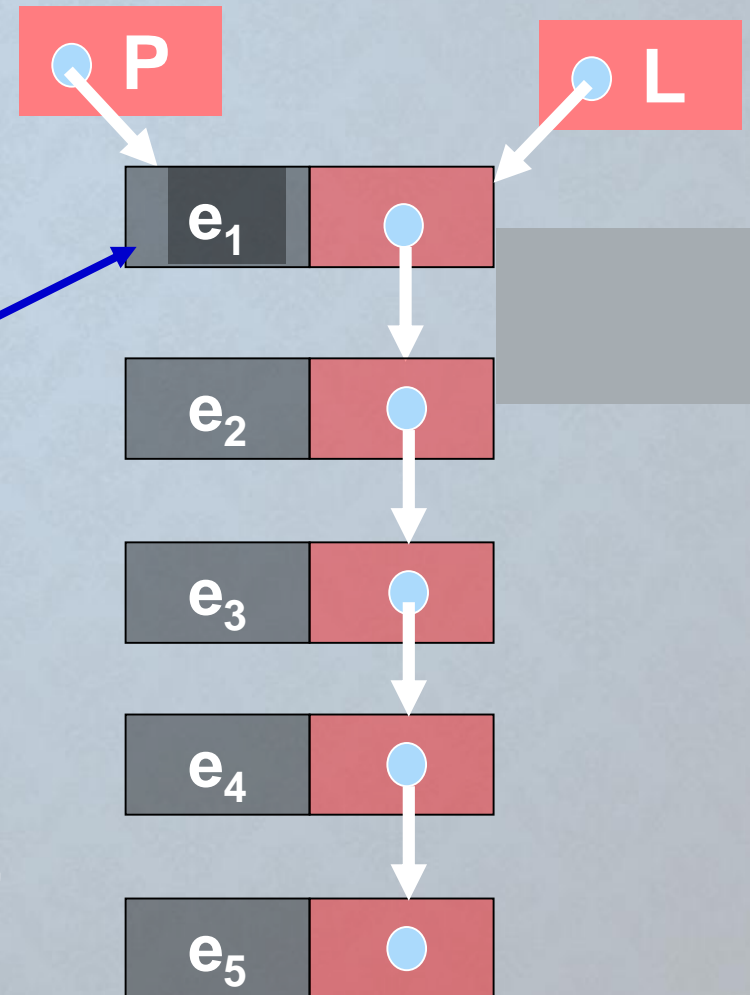
Khai báo con trỏ P: **TRO P;**

Cấp phát bộ nhớ cho con trỏ P:  
**P = new Node;**

Đưa dữ liệu vào node mới:  
**P->infor = elem;**

next của node mới trỏ vào phần tử đầu của danh sách: **P->next = L;**

L trỏ vào node mới: **L = P;**



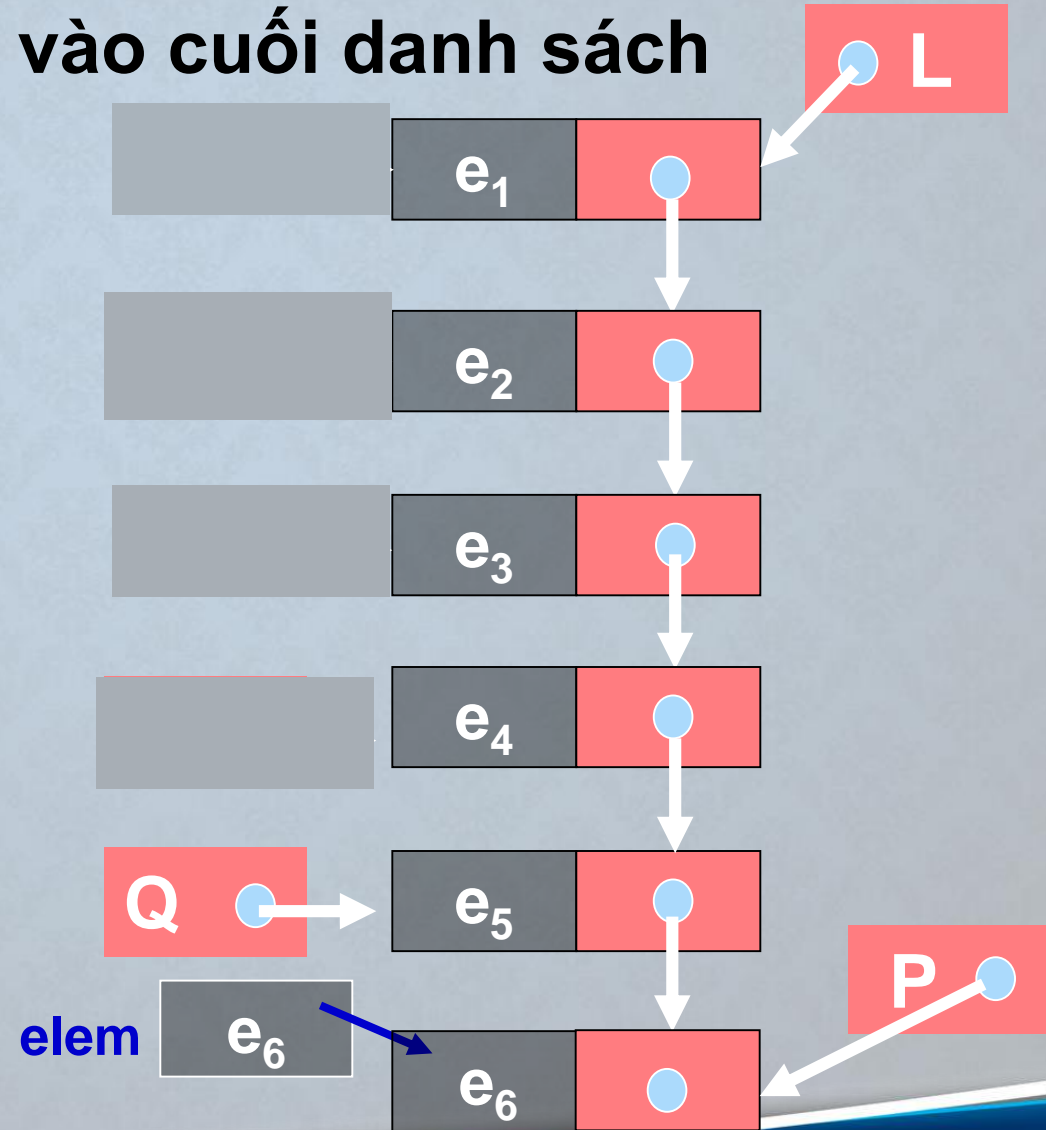
## Chèn một nút vào đầu danh sách (tt)

```
void firstAdd(TRO &L, DataType elem)
{
    TRO P;
    P = new Node;
    P->infor = elem;
    P->next = L;
    L = P;
}
```



#### 4.4.4.6. Chèn một nút vào cuối danh sách

1. Khởi tạo một node mới và đưa dữ liệu vào node mới.
2. Đưa con trỏ Q tìm đến cuối danh sách.
3. Nối node cuối với node mới



## Chèn một nút vào cuối danh sách (tt)

```
void add(TRO &L, DataType elem){  
    TRO P,Q;  
    P = new Node; P->infor = elem;  
    P->next = NULL;  
    if (empty(L)) L = P;  
    else {  
        Q = L;  
        while (Q->next != NULL)  
            Q = Q->next;  
        Q->next = P;  
    }  
}
```

#### 4.4.4.7. Chèn một nút vào sau nút trở bởi Q

Danh sách có phần tử đầu tiên  
được trở bởi con trỏ L

Q trở vào node mà node mới  
được bổ sung vào sau nó

Dữ liệu lưu trong biến elem

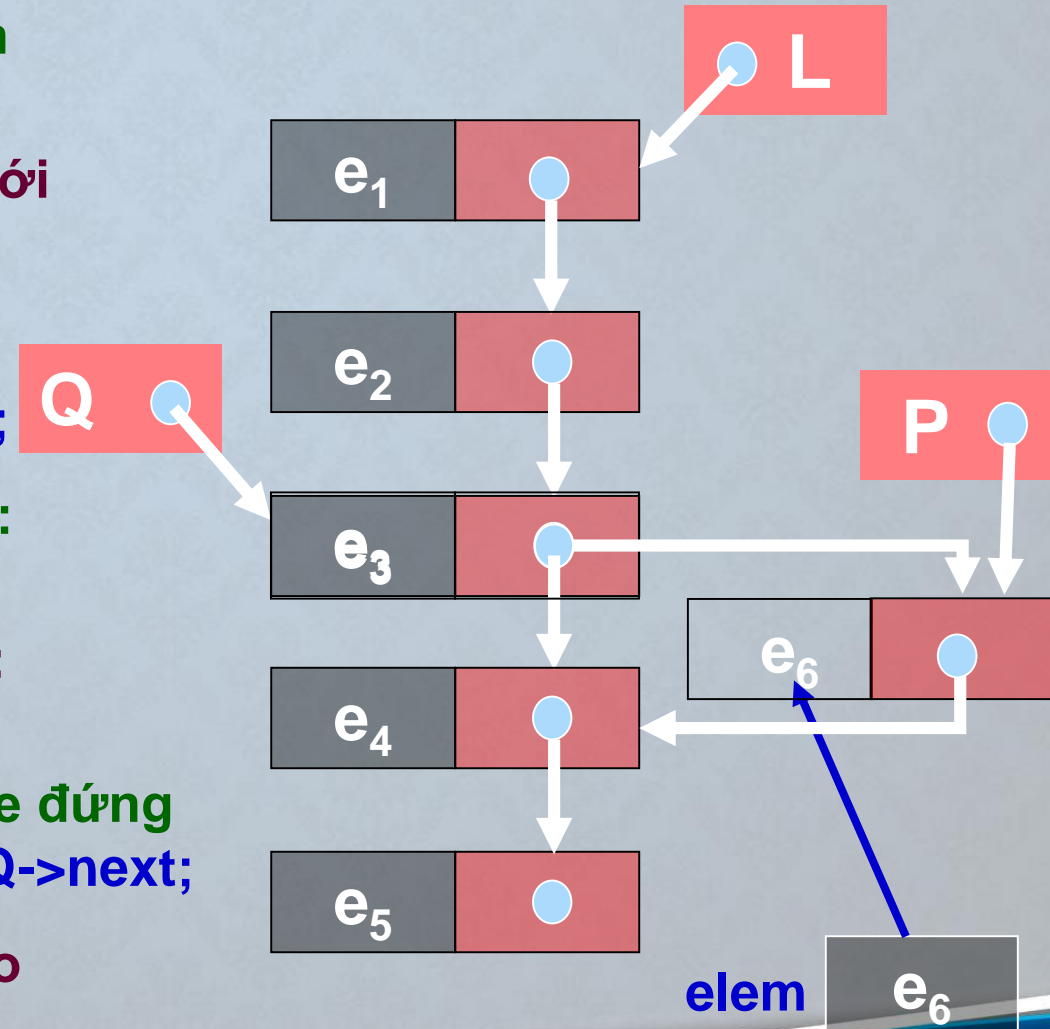
Khai báo con trỏ P: `TRO P;`

Cấp phát bộ nhớ cho con trỏ P:  
`P = new Node;`

Đưa dữ liệu vào node mới:  
`P -> infor = elem;`

next của node mới trở vào node đứng  
sau node trở bởi Q: `P -> next = Q -> next;`

next của node trở bởi Q trở vào  
node mới: `Q -> next = P;`



## Chèn một nút vào sau nút trả bởi Q (tt)

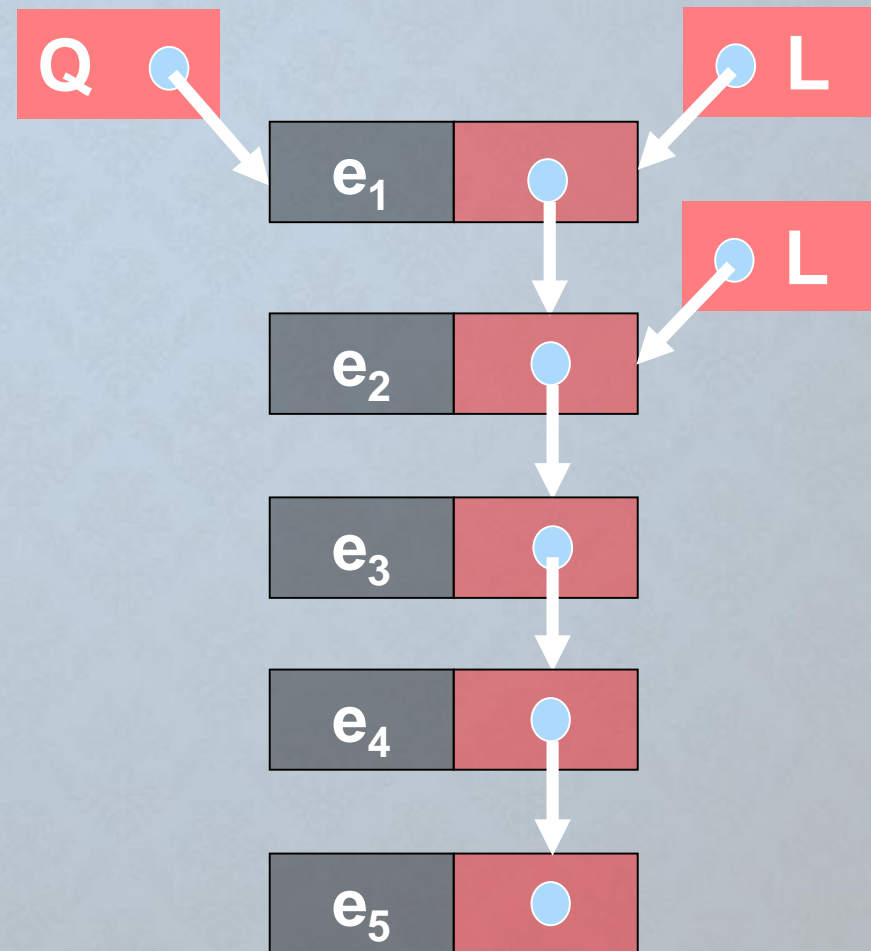
```
void insert(TRO &L, TRO Q, DataType elem)
{
    TRO P;
    P = new Node;
    P->infor = elem;
    P->next = Q->next;
    Q->next = P;
}
```

Hàm **insert** cũng thỏa mãn nếu bổ sung phần tử vào cuối danh sách, khi đó Q trở vào node cuối danh sách



#### 4.4.4.8. Xóa nút đầu tiên trong danh sách

1. Khai báo con trỏ Q: **TRO Q;**
2. Cho Q trở vào node đầu tiên:  
**Q = L;**
3. Chuyển L xuống node thứ 2:  
**L = L->next;**
4. Xóa node trở bởi con trỏ Q:  
**delete Q;**



## Xóa nút đầu tiên trong danh sách (tt)

```
void firstDelete(TRO &L)
{
    TRO Q;
    Q = L;
    L = L->next;
    delete Q;
}
```

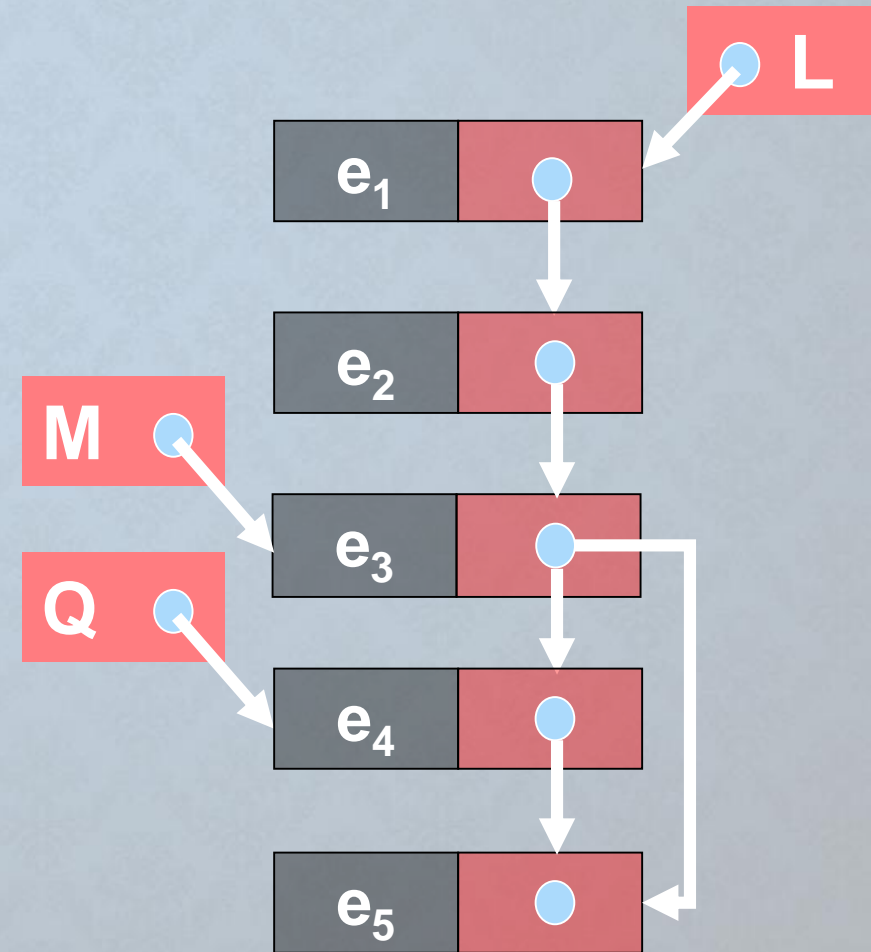
#### 4.4.4.9. Xóa nút đứng sau nút trở bởi con trỏ M

1. Khai báo con trỏ Q: **TRO Q;**

2. Cho Q trở vào node ở sau node trở bởi M: **Q = M->next;**

3. next của M trở vào node sau node trở bởi Q: **M->next = Q->next;**

4. Xóa node trở bởi con trỏ Q: **delete Q;**



## Xóa nút đứng sau nút trở bởi con trỏ M (tt)

```
void after_Delete(TRO &L, TRO M)
{
    TRO Q;
    Q = M->next;
    M->next = Q->next;
    delete Q;
}
```



## 4.4.5. Tạo một danh sách mới

**Xuất phát từ một danh sách rỗng: creat(L);**

1. Khai báo 2 con trỏ P, Q và biến elem: **TRO P, Q; DataType elem;**

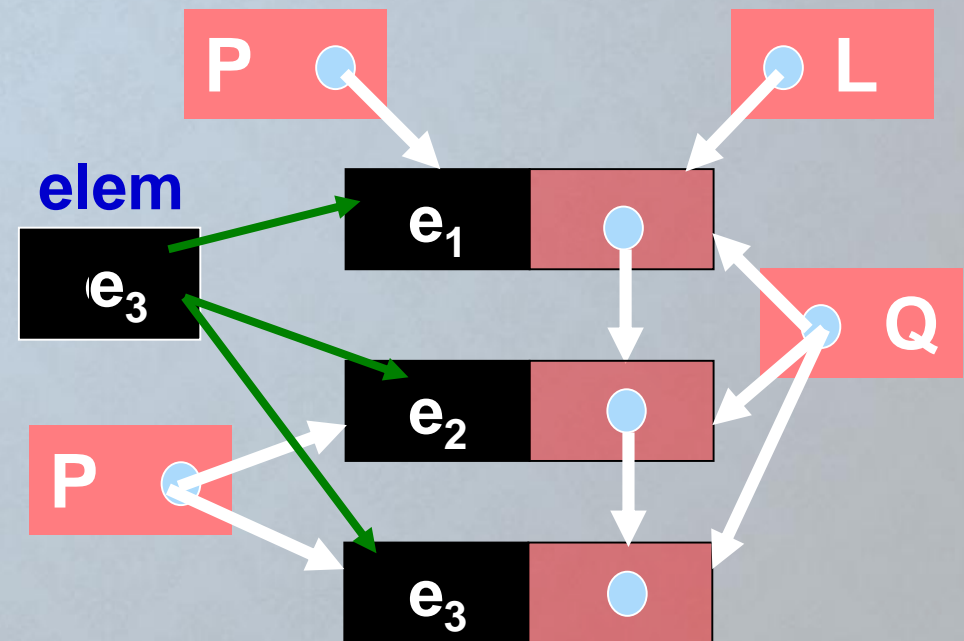
2. Nhập dữ liệu cho biến elem;

3. Cấp phát bộ nhớ cho con trỏ P và đưa dữ liệu vào chỗ nhớ đó, đồng thời **P->next=NULL;**

4. Nếu L=NULL thì L trở vào P Ngược lại next của node trỏ bởi Q trở vào node mới

5. Cho Q trở vào node mới

6. Nếu thỏa mãn điều kiện nhập tiếp thì lặp lại bước 2, ngược lại kết thúc



## Tạo một danh sách mới (tt)

```
void input_List(TRO &L){  
    TRO P, Q; DataType elem; char tieptuc; creat(L);  
    do{  
        input(elem);  
        P = new Node;  
        P->infor = elem; P->next = NULL;    if (L == NULL)  
        { L = P; }  
        else { Q->next = P; }  
        Q = P;  
        cout<<"Co nhap nua khong(C/K)?:";  
        cin>>tieptuc;  
    }while (toupper(tieptuc) == 'C');  
}
```

## Bài tập 1

- Chương trình quản lý sinh viên (mã SV, họ tên, năm sinh, điểm tổng kết) bằng danh sách nối đơn với các chức năng
  - Tạo mới danh sách
  - Hiển thị danh sách
  - Xác định chiều dài danh sách
  - Tìm kiếm sinh viên theo mã và hiển thị thông tin của sinh viên nếu tìm thấy.

## Bài tập 2

- Chương trình quản lý sinh viên (mã SV, họ tên, năm sinh, điểm tổng kết) bằng danh sách nối đơn với các chức năng:
  - Nhập mới n phần tử cho danh sách
  - Hiển thị danh sách ra màn hình
  - Hiển thị danh sách sinh viên có điểm tổng kết từ 6.5 trở lên
  - Chèn một sinh viên mới vào danh sách theo vị trí k (k nhập từ bàn phím).



## Bài tập 3

- Cho danh sách sinh viên như bảng dưới đây

STT	Mã SV	Họ đệm	Tên	Giới tính	Năm sinh	Điểm TK
1	1001	Tran Van	Thanh	Nam	1997	7.5
2	1002	Nguyen Thi	Hong	Nu	1998	7.2
3	1003	Nguyen Van	Hung	Nam	1996	6.4
4	1004	Bui Thi	Bich	Nu	1998	8.6
5	1005	Duong Van	Giang	Nam	1997	6.8

## Bài tập 3

- Giả sử danh sách được lưu trữ trong bộ nhớ máy tính dưới dạng danh sách nối đơn.
- Thực hiện các yêu cầu sau với danh sách:
  - Mô tả cấu trúc dữ liệu của danh sách qua hình vẽ.
  - Khai báo cấu trúc dữ liệu của danh sách.
  - Mô tả thao tác xóa phần tử đầu tiên trong danh sách bằng hình vẽ.
  - Cài đặt hàm xóa phần tử đầu tiên trong danh sách.
  - Mô tả thao tác chèn sinh viên (1006, Le Thi, Doan, Nu, 1998, 7.6) vào vị trí thứ 3 trong danh sách.
  - Mô tả thao tác sắp xếp danh sách theo chiều tăng dần của tên sinh viên bằng phương pháp lựa chọn.
  - Cài đặt chương trình mô phỏng các thao tác trên.



## Bài tập 4

- Cho danh sách hàng hóa như bảng dưới đây:

STT	Mã hàng	Tên hàng	ĐV tính	Đơn giá	Số lượng	Thành tiền
1	2001	Vở	Quyển	5000	20	100000
2	2002	Bút chì	Cái	8000	50	400000
3	2003	Hộp bút	Chiếc	30000	10	300000
4	2004	Tẩy	Cái	10000	20	200000
5	2005	Mực	Lọ	12000	5	60000
6	2006	Thước kẻ	Chiếc	3000	15	45000



## Bài tập 4

- Giả sử danh sách được lưu trữ trong bộ nhớ máy tính dưới dạng danh sách nối đơn.
- Thực hiện các yêu cầu sau với danh sách: :
  - Mô tả cấu trúc dữ liệu của danh sách qua hình vẽ.
  - Khai báo cấu trúc dữ liệu của danh sách.
  - Mô tả thao tác xóa phần tử thứ 3 trong danh sách bằng hình vẽ.
  - Cài đặt hàm xóa phần tử thứ 3 trong danh sách.
  - Mô tả thao tác chèn hàng hóa (2007, Phấn, Hộp, 3000, 15, 45000) vào vị trí đầu tiên trong danh sách.
  - Mô tả thao tác sắp xếp danh sách theo chiều giảm dần của thành tiền bằng phương pháp nổi bọt.
  - Cài đặt chương trình mô phỏng các thao tác trên.





## Bài tập 5

- Chương trình quản lý sinh viên (mã SV, họ tên, năm sinh, điểm tổng kết) bằng danh sách nối đơn với các chức năng:
  - Nhập mới danh sách, việc nhập kết thúc khi mã sinh viên nhập vào là chuỗi rỗng.
  - Hiển thị danh sách sinh viên sinh năm 1998
  - Xóa phần tử đầu tiên trong danh sách, hiển thị lại danh sách
  - Xóa phần tử thứ 5 trong danh sách hiển thị lại danh sách.
  - Xóa sinh viên khi biết mã (nhập từ bàn phím)



## Bài tập 6

- Chương trình quản lý sinh viên (mã SV, họ tên, năm sinh, điểm tổng kết) bằng danh sách nối đơn với các chức năng:
  - Tạo mới danh sách 6 phần tử
  - Hiển thị danh sách
  - Thêm một phần tử vào đầu danh sách, hiển thị lại danh sách
  - Thêm một phần tử vào cuối danh sách, hiển thị lại danh sách
  - Thêm một phần tử vào vị trí thứ 5 trong danh sách, hiển thị lại danh sách.
  - Sắp xếp danh sách theo chiều tăng dần của điểm tổng kết.



# Thank you...!