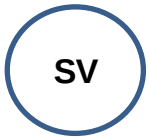
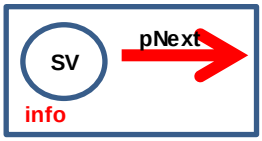
DANH SÁCH LIÊN KẾT ĐƠN_SINHVIEN (LIST<SinhVien>)

```
typedef struct SinhVien
{       char  maSV[11];
        char  hoten[51];
        float   dtb;
}SV;
//====================
```




```
typedef struct node
{       SV      info;
        node*  pNext;
}Node;
//====================
```




```
typedef struct list
{       node* pHead;
        node* pTail;
}List;
//====================
```

Bài 1: LIST <SV>

Gợi ý cần xây dựng các hàm:

```
void      init(List &l);
int       isEmpty(List l);
node*     createNode(SV X);
void      addHead(List &l, Node* p);
void      addTail(List &l, Node* p);
void      input(List &l);
void      output(List l);
…
```

Yêu cầu thực hành:

1/ Nhập số lượng Sinh viên
Tạo danh sách liên kết đơn chứa danh sách Sinh viên (***addHead / addTail***)
Xuất danh sách liên kết đơn chứa danh sách Sinh viên đó

2/ Đọc FILE dữ liệu chứa danh sách Sinh viên, đưa vào danh sách liên kết đơn (***addHead / addTail***)
Xuất danh sách liên kết đơn chứa danh sách Sinh viên đó

3/ Sắp xếp danh sách Sinh viên ***giảm dần*** theo ***hoten***, với thuật toán ***InterchangeSort*** **1đ**

4/ Sắp xếp danh sách Sinh viên ***tăng dần*** theo ***maSV***, với thuật toán ***SelectionSort***

5/ Nhập ***mã sinh viên*** cần tìm X: ?
Tìm tuyến tính ***mã sinh viên*** X, nếu tìm thấy xuất thông tin của Sinh viên đó. Ngược lại xuất thông báo.

6/ Nhập ***mã sinh viên*** cần tìm X: ?
Tìm tuyến tính ***mã sinh viên*** X, nếu tìm thấy nhập thông tin Sinh viên Y, chèn Sinh viên Y vào kế sau Sinh viên X. Ngược lại xuất thông báo.

7/ Xóa thông tin Sinh viên đầu danh sách

8/ Xóa thông tin Sinh viên cuối danh sách

9/ Xóa tất cả danh sách Sinh viên

10/ Nhập ***mã sinh viên*** cần xóa X: ?
nếu tìm thấy thì xóa thông tin của Sinh viên đó. Ngược lại xuất thông báo.

Còn tiếp…