BÀI TẬP VỀ DANH SÁCH LIÊN KẾT ĐƠN

Bài tập 01

1. Thêm phần tử vào đầu danh sách liên kết đơn
2. Thêm phần tử vào cuối danh sách liên kết đơn
3. Thêm phần tử vào vị trí bất kỳ trong danh sách liên kết đơn
4. Xóa phần tử đầu danh sách liên kết đơn
5. Xóa phần tử cuối danh sách liên kết đơn
6. Xóa phần tử tại vị trí bất kỳ trong danh sách liên kết đơn
7. Xóa tất cả các phần tử trong danh sách liên kết đơn
8. In các phần tử trong danh sách liên kết đơn

Bài tập 02

Đếm số node trong DSLK (Tìm độ dài DSLK)

Input: {1, 2, 2, 2, 3, 4, 4, 5}

Output: Sau khi thực hiện sẽ được kết quả là: 8

Bài tập 03

Đảo ngược DSLK

Input: {1, 2, 3, 4, 4, 5, 6, 7}

Output: Sau khi thực hiện sẽ được kết quả là: {7, 6, 5, 4, 3, 2, 1)

Bài tập 04

Tìm kiếm một phần tử 'x' ở bên trong một danh sách liên kết đơn.. Hàm này sẽ trả về true nếu x xuất hiện bên trong DSLK và trả về false trong trường hợp ngược lại.

Input: Tìm kiếm 2 trong DSLK{1, 2, 3, 4, 4, 5, 6, 7}

Output: Trong trường hợp này trả về true (Tìm thấy). Tuy nhiên, nếu tìm 15 thì trả về False (Không tìm thấy).

Bài tập 05

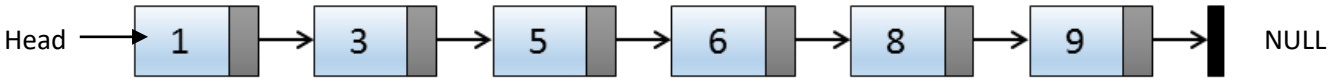
Cho một linked list được sắp xếp theo thứ tự tăng dần, hãy viết một hàm loại bỏ bất kỳ nút trùng lặp nào khỏi danh sách bằng cách duyệt qua danh sách chỉ một lần.

Input: {1, 2, 2, 2, 3, 4, 4, 5}

Output: Sau khi thực hiện sẽ được kết quả là: {1, 2, 3, 4, 5}

HƯỚNG DẪN THAM KHẢO

Bài tập 01




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

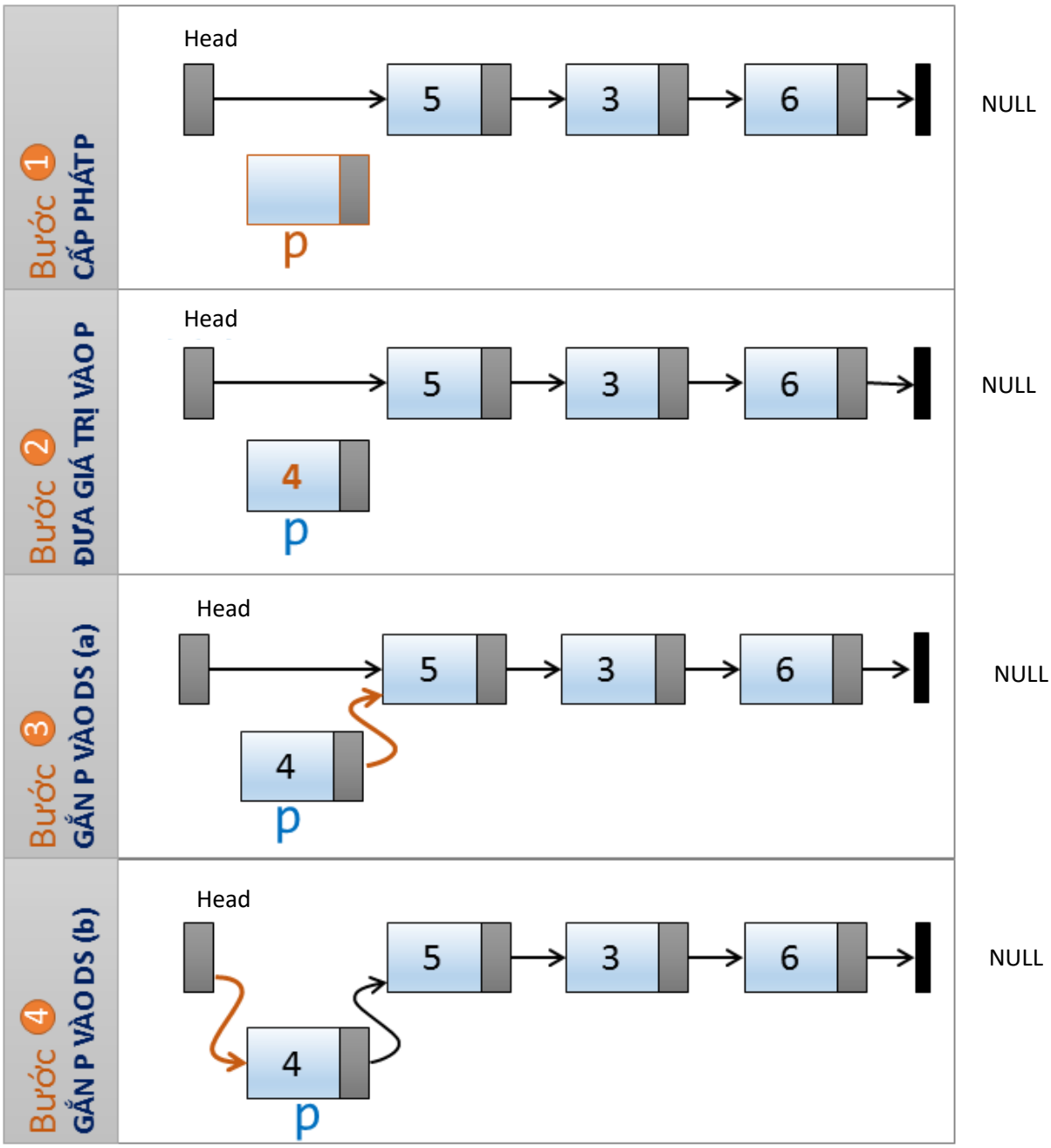
Cách 1 khai báo đơn thuần

```
struct node
{
   int data;
   struct node *next;
};
struct node *head = NULL;
```

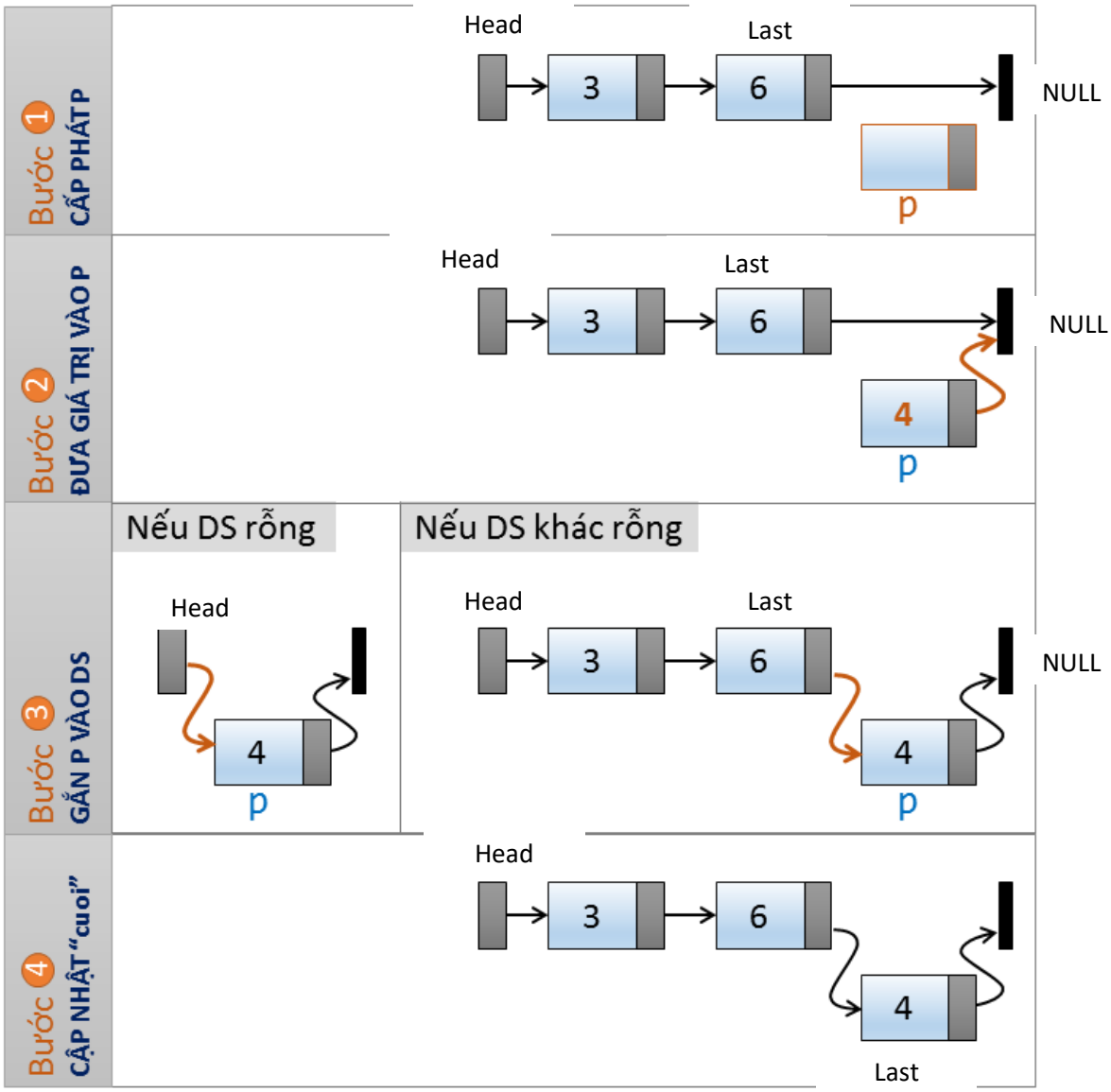
Cách 2 khai báo kiểu mới(Typedef)

```
typedef struct SV
{
   int data;
   struct SV *next;
};node
node *head = NULL;
```

THÊM PHẦN TỬ VÀO ĐẦU DANH SÁCH




```
void insert(int x) // thêm x vào đầu danh sách
{
   struct node *p = (struct node *)malloc(sizeof(struct node));
      // 1.khai báo và cấp phát vùng nhớ
   p->data = x;      // 2.Gán x cho p
   p->next = head;   // 3.Gắn p vào đầu con trỏ
   head = p;         // 4.Cập nhật con trỏ
}
```




```
void addlast(int x) // thêm x vào cuối danh sách
{
   struct node *p, *last;
   p = (struct node *)malloc(sizeof(struct node));
          // 1.khai báo và cấp phát vùng nhớ
   p->data = x;      // 2.Gán x cho p
   p->next = NULL;
   if (head == NULL)// 3.Gắn p vào danh sách
       head = p;
   else
      last = last->next;
   last = p;   // cập nhật con trỏ cuối
   }
}
```

```
void addAfter(int vt, int x) //chèn phần tử x tại vị trí vt
{
   struct node *p, *q;
   p = creatNode(x);   // tìm node có giá trị vt
   q = head;
   while (q != NULL && q->data != vt)
      q = q->next;
   if (q != NULL) // trong danh sách đã có phần tử vt thì thực
hiện câu lệnh sau
   {
      p->next = q->next;
      q->next = p;
   }
}
void deleteFirst()  // xóa node đầu tiên
{
   if (head != NULL)
   {
      struct node *p = head;
      head = p->next;
      p->next = NULL;
      free(p);
   }
}
void deleteLast() // xóa node cuối cùng
{
   if (head != NULL)
   {
     // xác định nút cuối cùng và nút trước của nút cuối cùng
      struct node *last = head;
      struct node *prev = NULL;
      while (last->next != NULL)
      {
         prev = last;
         last = last->next;
      }
      if (prev = NULL) // ds chỉ có 1 phần tử
          deleteFirst();

      else // xóa phần tử cuối cùng trong danh sách
      {
         prev->next = NULL;
         free(last);
      }
   }
}
```

```
void deleNode(int vt) // Xóa node tại vị trí vt
{
   if (head != NULL)
   {
      struct node *p = head;
      struct node *prev = NULL;
      while (p != NULL && p->data != vt) // p != NULL TRƯỜNG HỢP
chỉ có 1 phần tử....
      {
         prev = p;
         p = p->next;
      }
      if (p != NULL) // tìm thấy phần tử cần xóa
      {
         if (prev == NULL) // ds chỉ có 1 phần tử
             deleteFirst();
         else
         {
            prev->next = p->next;
            p->next = NULL;
            free(p);
         }
      }
   }
}
void deleteList() //Xóa tất cả các node
{
   struct node *p;

   while (head != NULL)
   {
      p = head;
      head = head->next;

      free(p);
   }

   printf("\nXOA THANH CONG TAT CA CAC NODE TRONG DANH SACH\n");
}
```

```
void printList()
{
    struct node *p = head;
    if (p == NULL)
        printf("\n-----DANH SACH RONG----\n\n");
    else
    {
        while (p != NULL)
        {
            printf("%d ->", p->data);
            p = p->next;
        }
    }
}
int main()
{
    // addlast(100);
    insert(5);
    insert(100);
    insert(40);
    printf("******* TRUOC KHI CHEN *******\n\n");
    printList();
    printf("\n\n************* SAU KHI CHEN *************\n\n");
    addAfter(100, 20);
    printList();
    // printf("\n\n************* SAU KHI XOA DAU
*************\n\n");
    // deleteFirst();
    // printList();
    // printf("\n\n************* SAU KHI XOA CUOI
*************\n\n");
    // deleteLast();
    // printList();
    // deleNode(325);
    // printf("\n\n************* XOA VI TRI BAT KY
*************\n\n");
    // printList();
    // printf("\n************* XOA HET *************\n");
    // deleteList();
    return 0;
}
```