1.Write a program to take age for passport registration portal: Sort the marks and arrange them in ascending order using bubble sort.

2.Write a program to implement a Singly Linked List for storing student PRN no.:Delete the PRN of admission cancelled student

1. Bubble Sort for Ages

Ugh bubble sort… the algorithm equivalent of walking when you could've taken an Uber.

Working:
- **You compare each pair of adjacent elements.**
- **If the left one is bigger, you swap their asses.**
- **After one full pass, the largest value floats to the end like an airhead in a pool.**
- **You repeat this for n-1 passes until everything is in ascending order.**
- **It's slow, basic, and still used because teachers love suffering.**

Theory points:
- **Time complexity worst case: $O(n^2)$**
- **Stable sorting technique**
- **Works by repeated swapping**
- **Good for small inputs; terrible for anything serious**
- **"Bubble" name because large elements rise upward**

You literally feed n ages, it sorts them in ascending order, prints them. Done.

---

2. Singly Linked List Deleting PRN

Now this one actually makes sense in real systems unlike bubble sort.

Working:
- **Each node stores a PRN and a pointer to the next student in the chain**
- **New student? You create a node and link it.**
- **Cancelled admission? You find that PRN node and remove it by reconnecting the previous node to the next one.**
- **If the deleted node is the first one, you just move the head to head->next.**
- **Memory of that node is freed so you don't create a digital ghost student.**

Theory points:

- **Linked List is a dynamic data structure**
- **Nodes stored non-contiguously**
- **Supports insertions & deletions efficiently**
- **Access is sequential**
- **Head pointer stores entry point**
- **Deletion needs three steps: search, unlink, free**

Your program:

- **Takes n PRNs**
- **Inserts them at the head one by one**
- **Takes a PRN to delete**
- **Searches for it**
- **Removes if found**
- **Displays final list**

1. Bubble sort for ages

```
#include <stdio.h>


int main() {
   int n, i, j, temp;
   scanf("%d", &n);
   int a[n];
   for(i=0;i<n;i++) scanf("%d",&a[i]);
   for(i=0;i<n-1;i++)
      for(j=0;j<n-i-1;j++)
         if(a[j] > a[j+1]) {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
         }
```

```
    for(i=0;i<n;i++) printf("%d ",a[i]);
    return 0;
}
```

2. Singly linked list delete PRN

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

struct node {
    int prn;
    struct node *next;
};

struct node* insert(struct node* head, int prn) {
    struct node* t = malloc(sizeof(struct node));
    t->prn = prn;
    t->next = head;
    return t;
}

struct node* delete(struct node* head, int prn) {
    struct node *p = head, *q = NULL;
    while(p && p->prn != prn) {
        q = p;
        p = p->next;
    }
    if(!p) return head;
    if(!q) head = p->next;
    else q->next = p->next;
```

```
    free(p);
    return head;
}

void display(struct node* head) {
    while(head) {
        printf("%d ", head->prn);
        head = head->next;
    }
}

int main() {
    int n, x, del;
    struct node* head = NULL;

    scanf("%d", &n);
    for(int i=0;i<n;i++) {
        scanf("%d",&x);
        head = insert(head, x);
    }

    scanf("%d",&del);
    head = delete(head, del);
    display(head);
    return 0;
}
```