카테고리 없음

자료구조 복습문제_

Lisa_Ha 2023. 9. 12. 02:09 수정 삭제

백_준 2346번 문제

문제

1번부터 N번까지 N개의 풍선이 원형으로 놓여 있고. i번 풍선의 오른쪽에는 i+1번 풍선이 있고, 왼쪽에는 i-1번 풍선이 있다. 단, 1번 풍선의 왼쪽에 N번 풍선이 있고, N번 풍선의 오른쪽에 1번 풍선이 있다. 각 풍선 안에는 종이가 하나 들어있고, 종이에는 -N보다 크거나 같고, N보다 작거나 같은 정수가 하나 적혀있다. 이 풍선들을 다음과 같은 규칙으로 터뜨린다.

우선, 제일 처음에는 1번 풍선을 터뜨린다. 다음에는 풍선 안에 있는 종이를 꺼내어 그 종이에 적혀있는 값만큼 이동하여 다음 풍선을 터뜨린다. 양수가 적혀 있을 경우에는 오른쪽으로, 음수가 적혀 있을 때는 왼쪽으로 이동한다. 이동할 때에는 이미 터진 풍선은 빼고 이동한다.

예를 들어 다섯 개의 풍선 안에 차례로 3, 2, 1, -3, -1이 적혀 있었다고 하자. 이 경우 3이 적혀 있는 1번 풍선, -3이 적혀 있는 4번 풍선, -1이 적혀 있는 5번 풍선, 1이 적혀 있는 3번 풍선, 2가 적혀 있는 2번 풍선의 순서대로 터지게 된다.

입력

첫째 줄에 자연수 N(1 ≤ N ≤ 1,000)이 주어진다. 다음 줄에는 차례로 각 풍선 안의 종이에 적혀 있는 수가 주어진다. 종이에 0은 적혀있지 않다.

출력

첫째 줄에 터진 풍선의 번호를 차례로 나열한다.

처음에는 단순 원형리스트 라고 생각 했는데, 문제를 자세히 읽어니 왼쪽으로도 이동해야 한다는 말이 있었다.
그래서 이중 원형 연결 리스트를 이용해서 문제를 풀어 보고자 한다.

처음에는 Node구조체를 선언하고, List 구조체 또한 선언하여 이중 연결리스트 처럼 구현 하려 했으나 이중 연결 리스트 사용 방법이 정확히 이해가 안되어 그냥 Node구조체에 리스트 번호 변수를 추가 하여 구현 하려 한다.

중간 중간 /* */ 로 둘러쌓여 있는 코드는 그저 구현을 하려다 실패하거나 검토를 위한 코드이니 가볍게 무시 하면 된다.

나의 코드의 계획은 종이에 적힌 N만큼 이동을 하고, 이동한 노드를 삭제하는데 그때 해당 노드에 적힌 N1 종이를 int변수에 저장하여 다음 for문 차례일때 다시 이동하는 값으로 넣는다.

근데 문제가 생겼다. 사실 이미 중간 점검때 발견한 문제 였는데 원인도 모르고 해결 방법도 몰라서 냅둔 문제이다.
내가 원하는 방향과 반대 방향으로 이동한다...

원인은 아직 찾지 못하였다. N = N -> R 이면 오른쪽에 있는(next) 노드로 이동하는게 정상이 아닌가?

최종 코드까지 작성하고 출력과 계산 또한 정상 작동 하지만 방향만 반대이다.

닫기

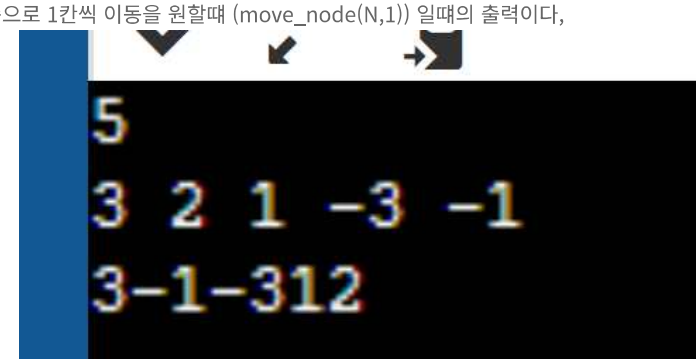
```c
#include<stdio.h>
#include<string.h>

typedef struct Node {
    int element;
    struct Node *L;
    struct Node *R;
    int count;
}Node;

/*
typedef struct list {
    int count;
    Node *list;
```

```c
}list;*/

void init(Node *N) {
    N->L = N;
    N->R = N;
}

Node *move_node(Node *N,int jong_i) {
    if (jong_i==0)return N;

    int i;
    //int cnt = N->element;
    if (jong_i < 0) {
        for (i = 0; i < -jong_i; i++) {
            N = N->L;
        }
    }
    else {
        for (i = 0; i < jong_i; i++) {
            N = N->R;
        }
    }
    return N;
}

void create_new(Node *before, int new_element, int cnt) {
    if (cnt == 1) {
        before->element = new_element;
        before->L = before;
        before->R = before;
        before->count = 1;
    }
    Node *new_node = (Node*)malloc(sizeof(Node));
    new_node->element = new_element;
    new_node->L = before;
    new_node->R = before->R;

    new_node->count = cnt;

    before->R->L = new_node;
    before->R = new_node;
}
/*
int pop_pungsun(Node *head, Node *Delete) {
    if (Delete == head) return;
    Delete->L->R = Delete->R;
    Delete->R->L = Delete->L;
    int re = Delete->element;
    free(Delete);
}*/
int pop_pungsun(Node *head,Node *Delete) {

    int re = Delete->element;

    if (Delete == head) {
        printf("%d", Delete->count);
        return;
    }
    printf("%d", Delete->count);

    Delete->L->R = Delete->R;
    Delete->R->L = Delete->L;

    free(Delete);

    return re;
}

int main() {
    Node *head = (Node*)malloc(sizeof(Node));
    init(head);

    int len;
    scanf("%d", &len);
    getchar();

    int ele;
    int jong_i=0;

    /*--노드 입력받기--*/
    for (int i = 0; i < len; i++) {
```

```
        scanf("%d", &ele);
        getchar();
        create_new(head, ele,i + 1);
    }
    /*
    for (int i = 0; i < len; i++) {
        printf("%d", head->element);
        head=move_node(head, 1);
    }*/


    Node *stm = (Node*)malloc(sizeof(Node));
    stm = head;
    /*--풍선 터트리기--*/
    for (int i = 0; i < len; i++) {
        stm = move_node(head,jong_i);
        jong_i = pop_pungsun(head,stm);
    }
    return 0;
}
```

아직도 왜 반대로 연산이 되는지 모르겠다.

우선은 지금 상태에서 오른쪽으로 1칸씩 이동을 원할때 (move_node(N,1)) 일때의 출력이다,



음..? 뭔가 코드를 계속 수정했더니 돌아 올 수 없게 되었다.

뭔가 더 난잡해졌다. 그냥 다시 이중 연결 리스트 (LINK)구조체를 사용하는 방법으로 돌아가야겠다.

새로 고친 코드

이동하는 함수를 없애고 삭제 하면서 바로 해당 값 만큼 head를 이동하도록 수정하였다.
하지만 여전히 원하는 값은 나오지 않는다...

닫기

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

typedef struct Node {
    int element;
    struct Node* L;
    struct Node* R;
    int count;
}Node;

```

```
12
13  typedef struct list {
14      int count;
15      Node *link;
16  }list;
17
18  /*
19  Node* move_node(Node* N, int jong_i) {
20      if (jong_i == 0)return N;
21
22      int i;
23      //int cnt = N->element;
24      if (jong_i < 0) {
25          for (i = 0; i < -jong_i; i++) {
26              N = N->L;
27          }
28      }
29      else {
30          for (i = 0; i < jong_i; i++) {
31              N = N->R;
32          }
33      }
34      return N;
35  }*/
36
37  void create_new(list *head, int new_element, int cnt) {
38      Node* new_node = (Node*)malloc(sizeof(Node));
39      new_node->element = new_element;
40      new_node->count = cnt;
41
42
43      if (cnt == 1) {//리스트가 공백일때, 즉 첫 노드의 값을 입력 받았을 때.
44          head->link = new_node;
45      }
46      else {
47          Node *before = (Node*)malloc(sizeof(Node));
48          before = head->link;//즉, 첫번째 노드.
49
50          for (int i = 1; i < cnt - 1; i++) {//새로운 노드의 직전, 리스트의 마지막 노드
51              before = before->R;
52          }
53          before->R = new_node;
54
55          new_node->L = before;
56
57          head->link->L = new_node; //head->link 는 첫번째 노드를 말한다.
58          new_node->R = head->link;
59
60      }
61      head->count = cnt; //Node의 count와 list의 count는 같은듯 보이지만 나중에는 다름에 주의.
62  }
63
64  /*
65  int pop_pungsun(Node *head, Node *Delete) {
66      if (Delete == head) return;
67      Delete->L->R = Delete->R;
68      Delete->R->L = Delete->L;
69      int re = Delete->element;
70      free(Delete);
71  }*/
72
73  void pop_pungsun(list *head) {
74
75      int re = head->link->count;
76      int jong_i = head->link->element;
77
78      if (head->count==1) {//하나의 노드만 남아있을 경우(마지막 풍선)
79          printf("%d", re);
80          return;
81      }
82      printf("%d", re);
83
84      Node* Delete = (Node*)malloc(sizeof(Node));
85      Delete = head->link;//즉, 삭제하고자 하는 노드 (첫번째 시도이서는 head의 위치가 변하지 않았으므로 첫번째 노드가 Delete
86
87      Delete->L->R = Delete->R;
88      Delete->R->L = Delete->L;
89
90      free(Delete);
91      head->count--;
92
```

```
    if (jong_i < 0) {
        for (int i = 0; i <jong_i*(-1); i++) {
            head->link = head->link->L;
        }
    }
    else {
        for (int i = 0; i < jong_i; i++) {
            head->link = head->link->R;
        }
    }

}

int main() {
    list *head = (list*)malloc(sizeof(list));
    head->link = NULL;
    head->count = 0;//초기화

    int len;
    scanf("%d", &len);
    getchar();

    head->count = len;
    int ele;
    int jong_i = 0;

    /*--노드 입력받기--*/
    for (int i = 0; i < len; i++) {
        scanf("%d", &ele);
        getchar();
        create_new(head, ele, i + 1);
    }

    /* 이미 초기화 했기 때문에 필요 없음.
    for (int i = 0; i < len; i++) {
        printf("%d", head->element);
        head=move_node(head, 1);
    }*/

    /*--풍선 터트리기--*/
    for (int i = 0; i < len; i++) {
        pop_pungsun(head);
    }

    free(head->link);
    free(head);

    return 0;
}
```

공감

댓글 0

Lisa_Ha

내용을 입력하세요.

등록