**프로그램 보고서 #5**

1612169 배현진

(1) 사용한 각 함수에 대한 설명

***- 문항별로 공통으로 사용한 함수***

1. void NodeInit(List \*plist) {

plist->head = NULL;

plist->tail = NULL;

}

- Input: 노드를 삽입할 리스트인 \*plist

- Output: void

- 알고리즘 설명: 리스트를 사용하기 전 NULL로 초기화 시켜주는 함수이다.

2. void print(List \*plist) {

node \*temp = plist->head;

int i;

printf("전체 리스트: ");

for (i = 0; i<37; i++) {

if (i == 36)

printf("%s", temp->data);

else

printf("%s -> ", temp->data);

temp = temp->next;

}

printf("\n");

}

- Input: 출력할 리스트인 \*plist

- Output: void

- 알고리즘 설명:

(1) temp 노드를 새로 만들어 매개 변수로 받은 리스트의 head 노드로 초기화한다.

(2) for문을 돌며 temp의 데이터를 출력한다. 그런 다음 temp를 temp의 다음 노드로 만들어준다.

***- 5-1번***

3. void SingleNodeInsert(List \*plist, char \*data) {

node \*newnode = (node\*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

printf("삽입중: %s\n", newnode->data);

if (plist->head == NULL)

plist->head = newnode;

else

plist->tail->next = newnode;

plist->tail = newnode;

}

- Input: 노드를 삽입할 \*plist와 노드에 들어갈 값인 data

- Output: void

- 알고리즘 설명:

(1) 새로은 newnode 노드를 만들고 초기화 시킨다. 그리고 newnode의 data를 출력한다.

(2) 만약 plist의 head가 null이라면 newnode를 head로 만들어준다.

(3) 그렇지 않다면 plist의 tail 다음에 newnode를 연결시켜준다.

(4) newnode를 tail로 지정한다.

4. void makeSingleCircular(List \*plist) {

plist->tail->next = plist->head;

}

- Input: Circular 리스트로 만들 \*plist

- Output: void

- 알고리즘 설명: 매개변수로 받은 리스트의 head를 tail의 다음 노드로 지정해준다.

***- 5-2번***

5. void SingleNodeDelete(List \*plist, int ranNum) {

node \*temp, \*temp2;

int i, j;

temp = plist->head;

temp2 = plist->head;

printf("skip number: %d\n", ranNum);

for (i = 1; i < 35; i++) {

for (j = 1; j < ranNum; j++) {

temp = temp->next;

}

temp2 = temp->next;

printf("%d단계 deleted node: %s \n", i, temp2->data);

temp->next = temp2->next;

}

printf("Survivers: ");

for (i = 1; i < 4; i++) {

printf("%s ", temp->data);

temp = temp->next;

}

}

- Input: 출력할 리스트인 \*plist, 랜덤으로 뽑은 skip number인 ranNum

- Output: void

- 알고리즘 설명:

(1) temp, temp2 노드를 새로 만들어 매개 변수로 받은 리스트의 head 노드로 초기화한다.

(2) skip number를 출력한다.

(3) for문을 돌며 skip number 만큼 사람을 건너 뛰며 리스트에서 삭제를 한다. 34번 동안 for문을 돌면서 현재 노드인 temp에서 ranNum만큼 노드를 이동한 뒤 temp의 다음 노드의 다음 노드를 temp와 연결한다. 즉 temp 노드의 다음 노드를 삭제한다.

(4) for문을 다 돌고 난 뒤, 생존자 3명을 출력한다.

***- 5-3번***

6. void DoubleNodeInsert(List \*plist, char \*data) {

node \*newnode = (node\*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

newnode->prev = NULL;

printf("삽입중: %s\n", newnode->data);

if (plist->head == NULL)

plist->head = newnode;

else {

plist->tail->next = newnode;

newnode->prev = plist->tail;

}

plist->tail = newnode;

}

- Input: 노드를 삽입할 \*plist와 노드에 들어갈 값인 data

- Output: void

- 알고리즘 설명:

(1) 새로은 newnode 노드를 만들고 초기화 시킨다. 그리고 newnode의 data를 출력한다.

(2) 만약 plist의 head가 null이라면 newnode를 head로 만들어준다.

(3) 그렇지 않다면 plist의 tail 다음에 newnode를 연결시켜준다. 그리고 newnode의 이전을 tail과 연결한다.

(4) newnode를 tail로 지정한다.

7. void makeDoubleCircular(List \*plist) {

plist->tail->next = plist->head;

plist->head->prev = plist->tail;

}

- Input: Circular 리스트로 만들 \*plist

- Output: void

- 알고리즘 설명: 매개변수로 받은 리스트의 head를 tail의 다음 노드로 지정해준다. 그리고 tail을 head의 이전 노드로 지정해준다.

***- 5-4번***

8. void DoubleNodeDelete(List \*plist, int ranNum) {

node \*temp, \*temp2, \*temp3;

int i, j, z, x;

int RranNum, LranNum;

temp = plist->head;

temp2 = plist->head;

temp3 = plist->head;

RranNum = ranNum;

LranNum = ranNum - 1;

printf("right skip number: %d\n", RranNum);

printf("left skip number: %d\n", LranNum);

for (i = 1; i < 18; i++) {

//오른쪽으로 돌면서 삭제

for (j = 1; j < RranNum; j++) {

temp = temp->next;

}

temp2 = temp->next;

printf("%d단계 deleted node: %s \n", 2 \* i - 1, temp2->data);

temp->next = temp2->next;

temp2->next->prev = temp;

// 왼쪽으로 돌면서 삭제

for (z = 1; z < LranNum; z++) {

temp = temp->prev;

}

temp3 = temp->prev;

printf("%d단계 deleted node: %s \n", 2 \* i, temp3->data);

temp->prev = temp3->prev;

temp3->prev->next = temp;

}

printf("Survivers: ");

for (x = 1; x < 4; x++) {

printf("%s ", temp->data);

temp = temp->next;

}

}

- Input: 출력할 리스트인 \*plist, 랜덤으로 뽑은 skip number인 ranNum

- Output: void

- 알고리즘 설명:

(1) temp, temp2, temp3 노드를 새로 만들어 매개 변수로 받은 리스트의 head 노드로 초기화한다.

(2) 오른쪽 skip number와 왼쪽 skip number를 출력한다.

(3) for문을 돌며 skip number 만큼 사람을 건너 뛰며 리스트에서 삭제를 한다. 17번 동안 for문을 돌면서 오른쪽과 왼쪽으로 도는 연산을 차례로 수행한다. 일단 현재 노드인 temp에서 RranNum만큼 오른쪽으로 노드를 이동한 뒤 temp의 다음 노드의 다음 노드를 temp와 연결한다. 즉 temp 노드의 다음 노드를 삭제한다. 그런 다음 현재 노드에서 LranNum만큼 왼쪽으로 노드를 이동한 뒤 temp의 이전 노드의 이전 노드를 temp와 연결한다. 즉 temp 노드의 이전 노드를 삭제한다.

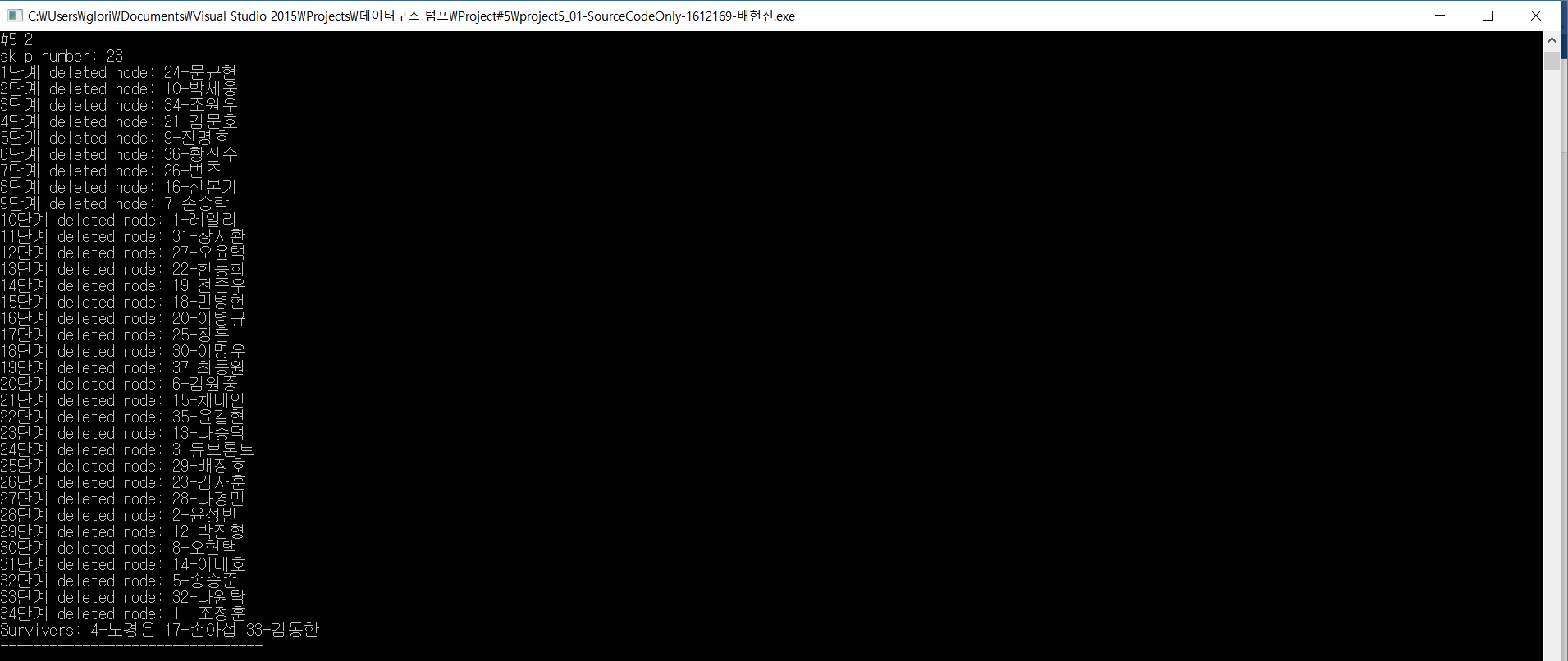
(4) for문을 다 돌고 난 뒤, 생존자 3명을 출력한다.

(2) 실행 결과 화면 단계별 Capture

***5-1. Singly Circular Linked List 입력 프로그램***



***5-2. Singly Circular Linked List: Skip Number를 뽑고 최종 3인 출력***



***5-3. Double Circular Linked List 입력 프로그램***



***5-4. Double Circular Linked List: Skip Number를 뽑고 최종 3인 출력***

