컴퓨터공학실험1 7주차 결과보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. 실험시간에 작성한 랭킹 시스템의 자료구조와 랭킹 시스템의 각 기능에 대한 알고리즘을 요약하여 기술하시오. 본인이 선택한 랭킹 시스템을 구현하기 위한 자료구조가 왜 효율적인지 시간 및 공간 복잡도를 통해 보이고, 설명하시오.

```
void createRankList() {
    // user code
    FILE *fp = fopen("rank.txt", "r");
    head_node.HEAD = NULL;
    head_node.n = 0; // 처음엔 0 명으로 초기화
    char inputname[NAMELEN];
    fscanf(fp, "%d", &(head_node.n)); //괄호...

    int inputscore;

    for (int i = 0; i < head_node.n; i++) {
        fscanf(fp, "%s%d", inputname, &inputscore);
        new_node(inputname, inputscore);
        //new_node 구현할 것!
    }
    fclose(fp);

}
```

rank.txt 파일을 파일 입출력으로 받아 저장된 랭크의 개수를 먼저 head_node.n에 저장한다. 이후 for문을 이용하여 반복해서 이름과 그에 상응하는 점수를 입력받고 new_node 함수를 호출하여 노드에 순차적으로 저장한다. (Linked List 이용, rank.txt 파일 안의 목록은 점수 순으로 순차적으로 정렬되어 있다 가정한다.)

```
void rank() {
    // user code
    char name[NAMELEN];
    clear();
    char opt;
    int inputdelete = 0;
    do {
        clear();
        printw("1. list ranks from X to Y\n");
        printw("2. list ranks by a specific name\n");
        printw("3. delete a specific rank\n");
        opt = wgetch(stdscr);
    } while (opt < '1' || opt > '3');

    echo(); //...
    int X, Y = -1; //초기화를 음수

    switch (opt) {
        case '1':
            printw("X: ");
            scanw("%d", &X);
```

```
            printw("Y: ");
            scanw("%d", &Y);
            if (X < 0) X = 1;
            if (Y < 0) Y = head_node.n;
            noecho(); //...
            //////////////////
            //printf("Hello\n\n");
            int index = 1;
            Node *idx = head_node.HEAD;
            printw("        name        |    score\n");
            printw("-----------------------------------------\n");
            //예외 처리를 할 것인가? ->조교님께 질문
            if (X > Y) {
                printw("search failure: the rank not in the list\n");
                break;
            }

            while (1) { //X 부터 Y 까지 반복 출력
                if (idx == NULL) break;
                if (X <= index && index <= Y) printw(" %-19s| %-12d\n", idx-
>name, idx->score);
                index++;
                idx = idx->link;
            }
            //////////////////
            break;
        case '2':
            echo();
            printw("input the name: ");
            scanw("%s", name);
            noecho();
            searchname(name);
            break;
        case '3':
            echo();
            printw("Input the rank: ");
            scanw("%d", &inputdelete);
            noecho();
            rank_delete(inputdelete);
            break;
    }
    noecho();
    getch();
}
```

메인 메뉴에서 2번 : rank를 선택한 이후로 나타나는 기능들을 다루는 함수이다.

“1. list ranks from X to Y”를 선택하면 rank의 X번째부터 Y번째까지의 사용자 데이터를 출력하는데, 그 전에 먼저 if문을 통해 예외 처리를 해 준다. 이후 while문을 이용해 인덱스를 1씩 증가시켜가며 X≤index≤Y인 동안 이름, 점수를 출력한다.

“2. list ranks by a specific name”를 선택하면 이름을 입력받고 searchname함수를 호출하여 입력받은 이름과 일치하는 정보를 찾는다.

“3. delete a specific rank”를 선택하면 숫자를 입력받고 그 숫자에 해당하는 번째의 rank를 삭제한다. 이 또한 rank_delete함수를 호출함으로써 구현한다.

(상기 언급한 호출되는 함수들에 대해서는 후술)

```
void writeRankFile() {
    // user code
    FILE *fp = fopen("rank.txt", "w");
    Node *insertnode = head_node.HEAD;

    fprintf(fp, "%d\n", head_node.n);
    while (1) { //NULL 만날 때까지 반복
        fprintf(fp, "%s %d\n", insertnode->name, insertnode->score);
        if (insertnode->link == NULL) break;
        insertnode = insertnode->link;
    }
    fclose(fp);
}
```

rank 메뉴 작업, 게임 종료 후 바뀐 정보를 rank.txt 파일에 다시 저장하는 함수이다. 맨 처음 노드의 개수(rank 목록의 개수)를 출력하고 그 다음부터 while문을 사용하여 이름과 그에 상응하는 점수를 순차적으로 출력한다. 이는 insertnode=insertnode->link와 같이 linked list를 순차적으로 탐색함으로써 이루어진다.

```
void newRank(int score) {
    // user code
    clear(); //빼먹지 말자...
    echo();
    char inputname[NAMELEN];
    printw("your name: ");
    scanw("%s", inputname);
    noecho();
    insert_node(inputname, score);

}
```

게임이 정상적으로 종료(Game Over!)되고 난 후 사용자 이름을 입력받고 점수를 저장한다. 알맞은 위치에 사용자 정보를 집어넣는 작업은 insert_node 함수에 의해 진행된다. (후술)

```
void new_node(char *name, int score) {
    Node *BACK = head_node.HEAD;
    Node *NEW_NODE;
    NEW_NODE = (Node *) malloc(sizeof(Node));
    NEW_NODE->name[0] = '\0'; // 없을 경우를 대비해 미리 초기화
    NEW_NODE->score = score;
    strcpy(NEW_NODE->name, name);
    NEW_NODE->link = NULL;

    if (head_node.HEAD == NULL) head_node.HEAD = NEW_NODE;
    else {
        while (1) {
            if (BACK->link == NULL) break;
            BACK = BACK->link;
        }

        BACK->link = NEW_NODE;
    }

}
```

새로운 노드를 만들고 이를 while문을 이용해 순차 탐색해 Linked List 맨 뒤에 저장한다.

```
void insert_node(char *name, int score) {
    //
    Node *first = head_node.HEAD;
    //
    Node *front = head_node.HEAD;
    Node *idxx = head_node.HEAD;
    Node *new;
    new = (Node *) malloc(sizeof(Node));
    new->name[0] = '\0';
    new->score = score;
    new->link = NULL;
    strcpy(new->name, name);
    head_node.n++;

    if (head_node.HEAD == NULL) head_node.HEAD = new; //No Node
    else if (head_node.HEAD->link == NULL) { //1 Node
        if ((new->score) > (head_node.HEAD->score)) {
            new->link = head_node.HEAD;
            head_node.HEAD = new;
        } else head_node.HEAD->link = new;
    } else { //2 or more Nodes
        ////인데, 첫번째보다 더 큰 경우->예외
        if ((new->score) > (head_node.HEAD->score)) {
            new->link = head_node.HEAD;
            head_node.HEAD = new;
            return;
        }
        ////
        while (1) {
            if ((new->score) > (idxx->score)) break;
            if (idxx->link == NULL) break;
            front = idxx;
            idxx = idxx->link;
        }
        if (front->score > new->score && new->score > idxx->score) {
            front->link = new;
            new->link = idxx;
        } else idxx->link = new;

    }

}
```

순차적으로 탐색하면서 새 노드를 알맞은 위치에 삽입하여 넣는 작업을 수행하는 함수이다. 이미 저장되어 있는 Linked List에 노드가 없을 때, 1개 있을 때, 2개 있을 때로 나누어 정리한다. 만약 노드가 없을 경우에는, HEAD를 새 노드로 설정하고, 노드가 1개 있을 경우에는 HEAD노드와의 비교를 진행한다. 2개 이상 있을 경우에는 먼저 HEAD노드보다 큰지 판별하여 예외 경우를 잡아주고, 이후 idxx=idxx->link의 방식을 사용해 while문 안에서 새 노드의 위치를 판별하여 잡아준다.

번외) 7주차 숙제 구현사항

```
void searchname(char *name) {
    bool success = false;
    Node *idx = head_node.HEAD;
    printw("        name        |    score\n");
    printw("------------------------------------\n");
    while (1) {
        if (!strcmp(idx->name, name)) { // 비교
            success = true;
            printw(" %-19s| %-12d\n", idx->name, idx->score);
        }
        if (idx->link == NULL) break;
        idx = idx->link; // 순차 탐색
    }
    if (!success) printw("search failure: no rank in the list\n");
}
```

strcmp 함수를 이용하여 입력받은 이름과 같은 rank를 찾아 출력한다. 이 또한 순차탐색이다.

```
void rank_delete(int deletenum) {
    Node *deletenode = head_node.HEAD;
    Node *front;
    int deletenode_num = 1;

    if (0 < deletenum && deletenum <= head_node.n) {
        if (deletenum != 1) {
            while (deletenode_num < deletenum) {
                front = deletenode;
                deletenode = deletenode->link;
                deletenode_num++;
            }
            front->link = deletenode->link;
            head_node.n--;
            free(deletenode);
            printw("result: the rank deleted\n");
        } else {
            head_node.HEAD = head_node.HEAD->link;
            head_node.n--;
            free(deletenode);
            printw("result: the rank deleted\n");
        }


    } else printw("search failure: no rank in the list\n");
}
```

입력받은 숫자에 해당하는 번째의 rank 정보를 삭제한다. if문을 통해 예외 경우를 걸러주고 난 후, while문을 이용한 순차 탐색으로 삭제할 노드를 찾는다. free 함수를 이용해 메모리에서 해제해주고, 링크를 재설정한다.

구현한 랭킹 시스템의 자료구조는 Linked List(링크드 리스트)이다. 시간복잡도는 노드(사용자)의 개수를 n이라 했을 때, O(n)이다. 공간복잡도도 O(n)이다. 각 함수마다 n 길이의 순차 탐색을 하는 선에서 연산이 끝나게 되는데, 만약 단순 배열을 자료구조로 채택했다면 정렬이나 중간 삽입에 있어서 더 많은 연산을 요하게 된다. 중간 삽입/삭제는 삽입/삭제된 원소 이후의 원소들을 모두 한 칸씩 옮겨주어야 하는 등 비효율성이 존재한다. 또한 만약 한꺼번에 정렬 연산을 해야 하는 경우에도 O(n) 만에 정렬하기 어렵다.

2. **본 실험 및 숙제를 통해 습득한 내용을 기술하시오.**

ncurses 라이브러리의 printw();, scanw();, echo();, clear();, getch(); 의 함수의 용례와 사용 방법을 확실히 익힐 수 있었다. 노드를 중간에 삽입하는 데 있어서 Linked List에 노드가 없는 경우, 1개 있는 경우, 2개 이상 있는 경우 등으로 나눠 삽입을 진행하는 방식도 익히게 되었다. (전체적으로 Linked List 삽입, 삭제, 정렬 연산에 있어 익숙해졌다.)

추가적으로 사용자 정의 함수를 여러 개 만들면서 Divide & Conquer (분할 정복) 큰 문제를 작은 단위로 쪼개어 구현하는 과정과 방법에 대해 좀 더 요령이 생겼다.