2020-2

데이터구조및실습

2차 Homework

- **소스코드 명: hw2_1_각자이름.cpp**

문제1

  - Binary tree를 사용하여 tree의 최대, 최소, key합계, child가 한 개인 노드의 수를 구하세요.
  - 조건
    - 이진트리(링크표현) 사용하기(이진탐색트리 아님)
    - 다음 각 사용자정의함수를 재귀적으로 정의하기
      - 최대key값
      - 최소key값
      - key값의 합계
      - child를 한 개 갖는 노드의 수
    - 힌트:
      - max의 초기값은 최대한 작게, min의 초기값은 최대한 크게 설정 (비교에 의해 정확한 값을 찾을 수 있도록)

문제1

- **소스코드 명: hw2_1_각자이름.cpp**

//tree정보(복사해서 사용)

```
TreeNode n0 = { 11, NULL, NULL };
TreeNode n1 = { 2, &n0, NULL };
TreeNode n2 = { 5, &n1, NULL };
TreeNode n3 = { 13, &n2, NULL };
TreeNode n4 = { 25, &n3, NULL };
TreeNode n5 = { 8, NULL, NULL };
TreeNode n6 = { 3, NULL, &n5 };
TreeNode n7 = { 21, &n4, &n6 };
TreeNode* exp = &n7;
```

```
Sum of keys in binary tree: 88
Num of one child nodes: 5
Max value: 25
Min value: 2
```

//tree정보(복사해서 사용)

```
TreeNode n0 = { 10, NULL, NULL };
TreeNode n1 = { 27, &n0, NULL };
TreeNode n2 = { 4, NULL, NULL };
TreeNode n3 = { 15, &n1, &n2 };
TreeNode n4 = { 16, NULL, NULL };
TreeNode n5 = { 25, NULL, NULL };
TreeNode n6 = { 7, &n4, &n5 };
TreeNode n7 = { 55, &n3, &n6 };
TreeNode n8 = { 40, NULL, NULL };
TreeNode n9 = { 36, NULL, &n8 };
TreeNode n10 = { 20, &n7, &n9 };
TreeNode* exp = &n10;
```

```
Sum of keys in binary tree: 255
Num of one child nodes: 2
Max value: 55
Min value: 4
```

- **소스코드 명: hw2_2_각자이름.cpp**

문제2

  - Binary Search Tree를 이용하여 과일명을 관리하는 코드를 구현하세요.
  - 조건
    - 5 입력시 종료, 그 이외의 경우 계속 반복
    - 다음 기능(사용자 정의 함수) 구현하기
      1. 새로운 단어입력(insert)
      2. 특정단어 삭제(delete)
      3. 특정단어 검색(search)
      4. 모든 단어 출력(print) – 정렬된 상태(Inorder)로
      5. 종료
    - 1~5이외의 입력시 “Wrong input”출력하고 계속 반복
    - 초기값을 초기화 함수 만들어서 입력하기
      - 초기값: apple, banana, blueberry, kiwi, orange, pear
    - 힌트:
      - TreeNode의 key는 문자배열(과일명)
      - 문자열관련 함수(strcmp, strcpy 등) 적절히 사용하기

• 소스코드 명: hw2_2_각자이름.cpp

문제2

```
* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):4
apple
banana
blueberry
kiwi
orange
pear

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):2
Enter a word to delete:pear

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):4
apple
banana
blueberry
kiwi
orange

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):1
Enter a word:strawberry

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):4
apple
banana
blueberry
kiwi
orange
strawberry

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):1
Enter a word:coconut
```

```
* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):4
apple
banana
blueberry
coconut
kiwi
orange
strawberry

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):3
Enter a word:orange
orange is found

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):2
Enter a word to delete:orange

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):4
apple
banana
blueberry
coconut
kiwi
strawberry

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):6
Wrong input

* Choose operation(1.insert 2.delete 3.search 4.print 5.exit):5
Program finished
```

- **소스코드 명: hw2_3_각자이름.cpp**

문제3

  - Dijkstra shortest path 코드를 링크표현법을 이용해서 구현하고 각 단계를 출력하세요.
  - 조건
    - 링크표현(vertex 포인터배열 및 연결된 vertex를 링크로 표현하기)
    - Graph 형태를 연결된 각 vertex번호 및 weight를 ()안에 써서 출력하는 함수 정의하여 호출하기
    - distance 및 found 배열 사용
    - 시작하고자 하는 vertex 번호 입력받기
    - 힌트
      - GraphNode에 weight 추가하기
      - 새로 found에 추가된 vertex u에 직접 연결된 노드 중 w가 있으면 (링크 따라가며 찾기),
        u~w간의 weight를 기존 distance값에 더한 결과와
        현재의 distance 값을 비교해서 distance 배열 update

```
typedef struct GraphNode {
    int vertex; // vertex 번호
    int weight;
    struct GraphNode* link;
} GraphNode;

typedef struct GraphType {
    int n;  // 정점의 개수
    GraphNode* adj_list[MAX_VERTICES];
} GraphType;
```

문제3

- 소스코드 명: hw2_3_각자이름.cpp

```
<Graph 형태>
vertex 0  -> 5(10)  -> 4(3)  -> 1(7)
vertex 1  -> 5(6)  -> 4(2)  -> 3(10)  -> 2(4)  -> 0(7)
vertex 2  -> 3(2)  -> 1(4)
vertex 3  -> 6(4)  -> 5(9)  -> 4(11)  -> 2(2)  -> 1(10)
vertex 4  -> 6(5)  -> 3(11)  -> 1(2)  -> 0(3)
vertex 5  -> 3(9)  -> 1(6)  -> 0(10)
vertex 6  -> 4(5)  -> 3(4)

시작 vertex번호: 0

<Shortest Path from vertex 0>
STEP 1:
 distance:  0  7  *  *  3 10  *
 found:  1  0  0  0  0  0  0

STEP 2:
 distance:  0  5  * 14  3 10  8
 found:  1  0  0  0  1  0  0

STEP 3:
 distance:  0  5  9 14  3 10  8
 found:  1  1  0  0  1  0  0

STEP 4:
 distance:  0  5  9 12  3 10  8
 found:  1  1  0  0  1  0  1

STEP 5:
 distance:  0  5  9 11  3 10  8
 found:  1  1  1  0  1  0  1

STEP 6:
 distance:  0  5  9 11  3 10  8
 found:  1  1  1  0  1  1  1

STEP 7:
 distance:  0  5  9 11  3 10  8
 found:  1  1  1  1  1  1  1
```

```
<Graph 형태>
vertex 0  -> 5(10)  -> 4(3)  -> 1(7)
vertex 1  -> 5(6)  -> 4(2)  -> 3(10)  -> 2(4)  -> 0(7)
vertex 2  -> 3(2)  -> 1(4)
vertex 3  -> 6(4)  -> 5(9)  -> 4(11)  -> 2(2)  -> 1(10)
vertex 4  -> 6(5)  -> 3(11)  -> 1(2)  -> 0(3)
vertex 5  -> 3(9)  -> 1(6)  -> 0(10)
vertex 6  -> 4(5)  -> 3(4)

시작 vertex번호: 3

<Shortest Path from vertex 3>
STEP 1:
 distance:  * 10  2  0 11  9  4
 found:  0  0  0  1  0  0  0

STEP 2:
 distance:  *  6  2  0 11  9  4
 found:  0  0  1  1  0  0  0

STEP 3:
 distance:  *  6  2  0  9  9  4
 found:  0  0  1  1  0  0  1

STEP 4:
 distance: 13  6  2  0  8  9  4
 found:  0  1  1  1  0  0  1

STEP 5:
 distance: 11  6  2  0  8  9  4
 found:  0  1  1  1  1  0  1

STEP 6:
 distance: 11  6  2  0  8  9  4
 found:  0  1  1  1  1  1  1

STEP 7:
 distance: 11  6  2  0  8  9  4
 found:  1  1  1  1  1  1  1
```

// 다음 입력자료 복사해서 사용하기

```
GraphType* g;
g = (GraphType*)malloc(sizeof(GraphType));
init(g);
for (int i = 0; i < 7; i++) insert_vertex(g, i);
insert_edge(g, 0, 1, 7);
insert_edge(g, 0, 4, 3);
insert_edge(g, 0, 5, 10);
insert_edge(g, 1, 0, 7);
insert_edge(g, 1, 2, 4);
insert_edge(g, 1, 3, 10);
insert_edge(g, 1, 4, 2);
insert_edge(g, 1, 5, 6);
insert_edge(g, 2, 1, 4);
insert_edge(g, 2, 3, 2);
insert_edge(g, 3, 1, 10);
insert_edge(g, 3, 2, 2);
insert_edge(g, 3, 4, 11);
insert_edge(g, 3, 5, 9);
insert_edge(g, 3, 6, 4);
insert_edge(g, 4, 0, 3);
insert_edge(g, 4, 1, 2);
insert_edge(g, 4, 3, 11);
insert_edge(g, 4, 6, 5);
insert_edge(g, 5, 0, 10);
insert_edge(g, 5, 1, 6);
insert_edge(g, 5, 3, 9);
insert_edge(g, 6, 3, 4);
insert_edge(g, 6, 4, 5);
```

- **소스코드 명: hw2_4_각자이름.cpp**

문제4

  - Quick sort에서 pivot을 median값으로 설정하고 결과를 비교하세요.
  - 조건
    - 리스트의 첫번째 수, 마지막 수, 중간에 위치한 수 중 median 값을 구하는 사용자 정의 함수(median of three)를 정의하기
    - 위 함수를 호출하여 pivot을 설정하고 pivot값을 단계별로 각각 출력하기
      - 단계별 리스트에 대해 첫번째 값이 pivot인 경우의 값(기존 예제)
      - 단계별 리스트에 대해 median of three 값이 pivot인 경우의 값(수정된 코드)

- 소스코드 명: hw2_4_각자이름.cpp

문제4

```
Input size of list(1~50):15
Input 15 numbers in the list
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29

<<정렬 이전 상태>>
   1    3    5    7    9   11   13   15   17   19   21   23   25   27   29
<Quick Sorting>
pivot: 1
pivot: 3
pivot: 5
pivot: 7
pivot: 9
pivot: 11
pivot: 13
pivot: 15
pivot: 17
pivot: 19
pivot: 21
pivot: 23
pivot: 25
pivot: 27
<Result>
   1    3    5    7    9   11   13   15   17   19   21   23   25   27   29

<Quick Sorting(Median of three)>
pivot: 15
pivot: 7
pivot: 3
pivot: 11
pivot: 23
pivot: 19
pivot: 27
<Result>
   1    3    5    7    9   11   13   15   17   19   21   23   25   27   29
```

Homework 결과 제출

1. 각 소스코드에 주석 30%이상 포함
2. 학과,학번,성명을 출력하는 함수를 호출하여 결과화면에 출력
3. 각 소스 코드와 결과화면 캡쳐한 그림 파일을 하나의 파일로 압축하여 제출

→ 압축파일명: HW2_각자이름.zip