Homework # 7

데이타구조론 (CSE2003-02)

HW#7 과제 안내

- 일정
  - 게시 : 5/13(목) 13:00
  - 제출 마감 : 5/20(목) 11:00 (delay 없음)
  - 채점 결과 확인 : 5/24(월)
  - 이의신청 마감 : 5/31(월) (이의신청 이메일 : greenlife124@yonsei.ac.kr, [데이타구조론HW#7 이의신청])

HW#7 과제 안내

- 설명
  - 모든 코드의 핵심 부분에는 comment를 달아 설명 할 것 (not option!!)
  - Compiler는 visual studio 2019 이상을 사용하여, HW#7_학번_이름 하나의 파일로 압축하여 제출 할 것
- HW#7_학번_이름
  - HW#7_1 > **Heap.c**, Heap.h, HeapMain.c
  - HW#7_2 > **BinaryTree.c**, BinaryTree.h, BSTMain.c

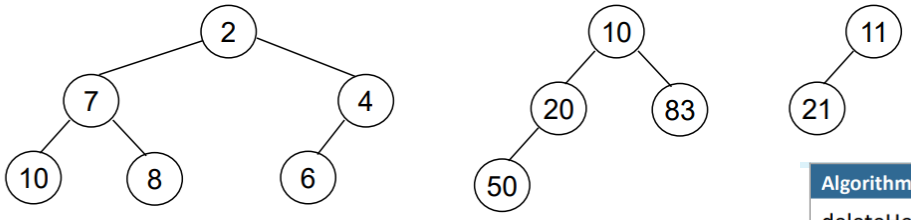
HW#7_1 최소 힙 구현

- 아래와 같이 실행되도록 **Heap.c** 작성
  - **insertHeap(), deleteHeap()** 작성
- Heap.h, HeapMain.c 제공
- 주의 사항
  - HW#7_1 폴더 안에, 주어진 3개의 코드 모두 존재 해야 함

```
1 8
2 13
3 10
4 20
5 15
6 19

(1) 5 삽입
1 5
2 13
3 8
4 20
5 15
6 19
7 10

(2) 16 삽입
1 5
2 13
3 8
4 16
5 15
6 19
7 10
8 20

(3) Heap 요소 차례로 삭제
1 5
2 8
3 10
4 13
5 15
6 16
7 19
8 20
```

HW#7_1 참고 자료

- Min Heap (최소 힙)



Algorithm void insertHeap(Heap* h, HData x)

```
insertHeap(h, x)
   if (n = MAX_SIZE) then overflow
   idx ← h.numOfData + 1
   while (idx ≠ 1) do
      if (x > h.heap[⌊idx/2⌋]) then
         h.heap[idx] ← h.heap[⌊idx/2⌋]
         idx ← ⌊idx/2⌋
      else break
   endwhile
   h.head[idx] ← x
   h.numOfData++
end insertHeap()
```

Algorithm void deleteHeap(Heap* h)

```
deleteHeap(h)
   if (isEmpty(h)) then underflow
   item ← h.heap[1]
   lastVal ← h.heap[h.numOfData]
   parentIdx ← 1
   childIdx ← 2

   while (childIdx ≤ h.numOfData) do
      if ((childIdx < h.numOfData) and (h.heap[childIdx] < h.heap[childIdx+1]) then
         childIdx++
      if (lastVal ≥ h.heap[childIdx]) then break
      else
         h.heap[parentIdx] ← h.heap[childIdx]
         parentIdx ← childIdx
         childIdx ← childIdx * 2
   endwhile

   h.heap[parentIdx] ← lastVal
   h.numOfData--
   return item
end deleteHeap()
```

HW#7_2 이진 탐색 트리 구현

- 실행되도록 **BinaryTree.c** 작성
  - **makeLSubtree(), makeRSubtree(), getLSubtree(), getRSubtree(), setData(), getData(), InorderTraverse(), searchBST(), insertBST(), deleteBST()**작성
- BinaryTree.h, BSTMain.c제공
- 주의 사항
  - HW#7_2 폴더 안에, 주어진 3개의 코드 모두 존재 해야 함

HW#7_2 참고 자료

Algorithm | **void insertBST(BinTree* bt, BTData x)**

```
insertBST(bt, x)
   p ← bt
   parent ← NULL    // 삽입할 노드의 부모 노드 저장

   while (p ≠ NULL) do
      if (x = p.key) then return
      if (x < p.key) then p ← p.left
      else p ← p.right
   end while

   newNode.key ← x
   newNode.left ← NULL
   newNode.right ← NULL

   if (bt = NULL) then bt ← newNode
   else if (x < parent.key) then parent.left ← newNode
   else parent.right ← newNode
end insertBST()
```

삽입할 노드 탐색

삽입할 노드 생성

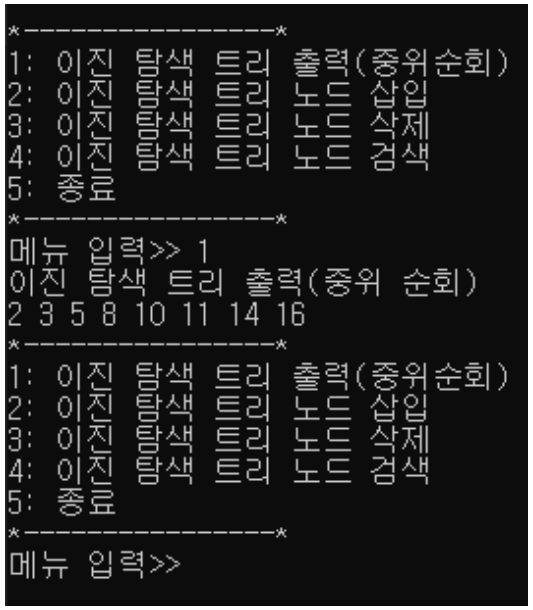
삽입 노드는 항상 단말 노드가 됨

삽입 노드를 부모 노드에 연결

Algorithm | **void deleteBST(BinTree* bt, BTData x)**

```
deleteBST(bt, x)
   p ← 삭제할 노드
   parent ← 삭제할 노드의 부모 노드

   // 삭제할 노드가 없는 경우
   if (p = NULL) then return

   // CASE 1: 삭제할 노드의 차수가 0인 경우
   if (p.left = NULL and p.right = NULL) then
      if (parent.left = p) then parent.left  ← NULL
      else parent.right ← NULL

   // CASE 2: 삭제할 노드의 차수가 1인 경우
   else if (p.left = NULL or p.right = NULL) then
      if (p.left ≠ NULL) then
         if (parent.left = p) then parent.left ← p.left
         else parent.right ← p.left
      else
         if (parent.left = p) then parent.left ← p.right
         else parent.right ← p.right

   // CASE 3: 삭제할 노드의 차수가 2인 경우
   else if (p.left ≠ NULL and p.right ≠ NULL) then
      q ← maxNode(p.left)  // 왼쪽 서브트리에서 후계자 노드 지정
      p.key ← q.key
      deleteBST(p.left, q.key)
end deleteBST()
```

HW#7_2 실행 화면

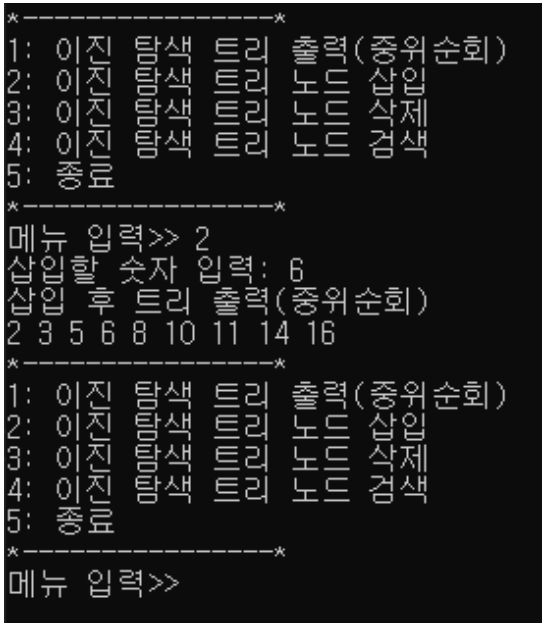


1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>> 1
이진 탐색 트리 출력(중위 순회)
2 3 5 8 10 11 14 16

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>>



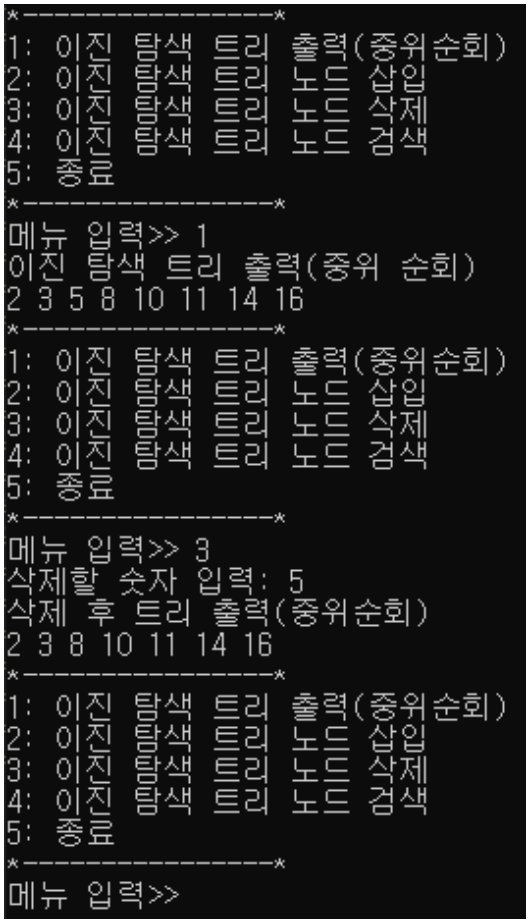
1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>> 2
삽입할 숫자 입력: 6
삽입 후 트리 출력(중위순회)
2 3 5 6 8 10 11 14 16

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>>

HW#7_2 실행 화면



1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

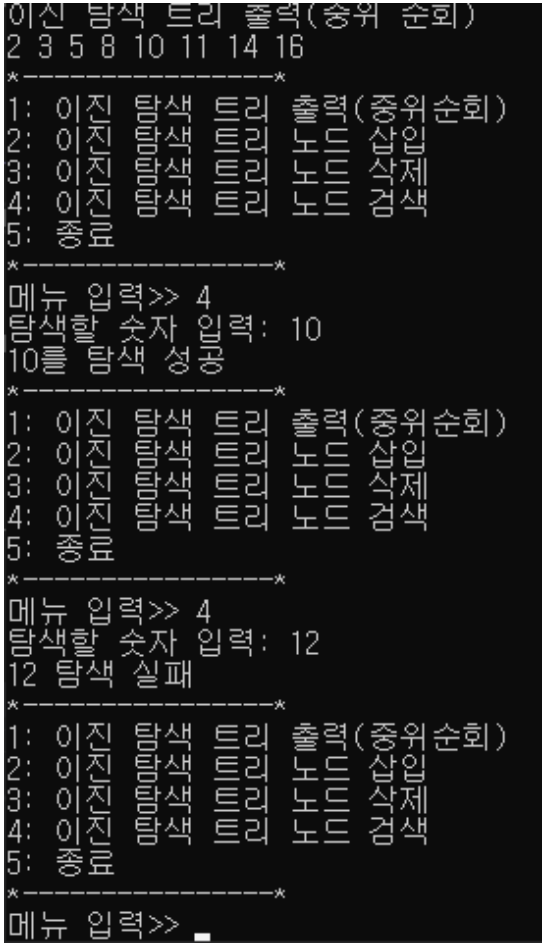
메뉴 입력>> 1
이진 탐색 트리 출력(중위 순회)
2 3 5 8 10 11 14 16

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>> 3
삭제할 숫자 입력: 5
삭제 후 트리 출력(중위순회)
2 3 8 10 11 14 16

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>>


이진 탐색 트리 출력(중위 순회)
2 3 5 8 10 11 14 16

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>> 4
탐색할 숫자 입력: 10
10를 탐색 성공

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>> 4
탐색할 숫자 입력: 12
12 탐색 실패

1: 이진 탐색 트리 출력(중위순회)
2: 이진 탐색 트리 노드 삽입
3: 이진 탐색 트리 노드 삭제
4: 이진 탐색 트리 노드 검색
5: 종료

메뉴 입력>>