실습시간은 40 분이며 그 전에 실습을 마친 학생은 답안지를 제출하시기 바랍니다. 답안지를 제출한 후에는 학생은 퇴실하거나 아직 마치지 못한 학생들을 도와줄 수 있습니다. 제한된 시간 안에 답안을 작성하지 못한 학생은 일단 답안지를 제출하고 미진한 답안은 금요일 (5/19) 16:50 까지 우정정보통신관 407A 로 제출하기 바랍니다.

Name: _____ Student ID: _____ Class: _____

담당교수: 김종규 _____

---

1. 다음 Tree search algorithm 을 iterative version 으로 변환하시오. 자료를 참조하거나 컴퓨터를 사용하여도 무방합니다.

```
TREE-SEARCH(x, k)
1  if x == NIL or k == x.key
2       return x
3  if k < x.key
4       return TREE-SEARCH(x.left, k)
5  else return TREE-SEARCH(x.right, k)
```

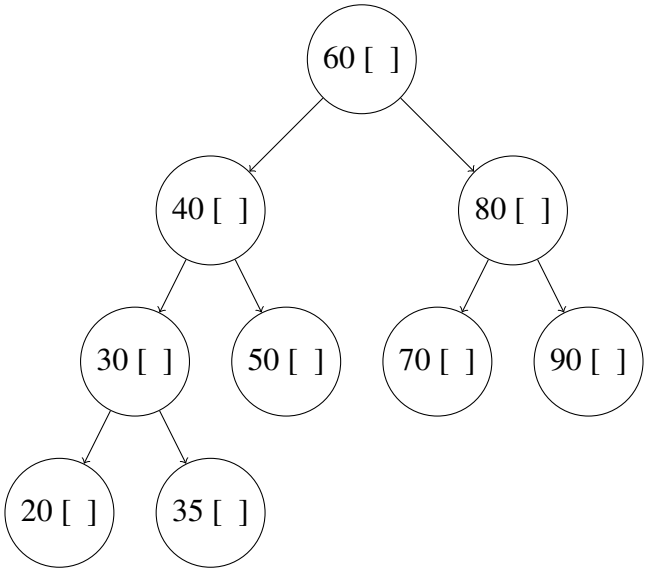
그림 1: Search

Answer:

```
Tree-search(x,k)
  while x != NIL and k != x.key
    if k < x.key
      x = x.left
```

```
else
  x = x.right
end while
```

2. 다음 binary tree 의 각 노드에 balance factor 를 계산하여 [ ] 안에 써넣으시오. 자료를 참조하거나 컴퓨터를 사용하여도 무방합니다.



Answer:
그림 2 참조

3. 다음 그림 3의 binary search tree 에 대한 질문에 답하시오. 자료를 참조하거나 컴퓨터를 사용하여도 무방합니다.

(a) 이 tree 의 inorder traveral 결과를 기술하시오.

Answer:
20 30 35 40 50 60 70 80 90

(b) 이 tree 의 preorder traveral 결과를 기술하시오.

Answer:
60 40 30 20 35 50 80 70 90

(c) 이 tree 의 postorder traveral 결과를 기술하시오.

Answer:
20 35 30 50 40 70 90 80 60

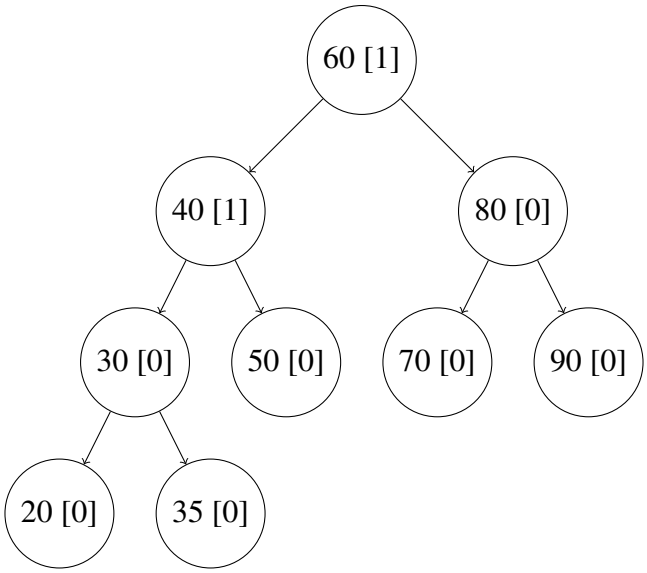


그림 2: Balance factor (bf)

(d) node 50 의 successor 를 구하시오.

Answer:

60

(e) node 70 의 predecessor 를 구하시오.

Answer:

60

(f) node 80 을 root 로 하는 subtree 에서 minimum 을 구하시오

Answer:

70

(g) 그림 3 의 tree 를 T 라고 할 때, 다음을 수행한 결과를 보이시오.

```
u = Tree-search(T, 40)
Right-rotate(T, u)
```

Answer:

그림 4 참조 .

(h) 그림 3 의 tree 를 T 라고 할 때, 다음을 수행한 결과를 보이시오.

```
u = Tree-search(T, 30)
v = Tree-search(T, 40)
Transplant(T, u, v)
```

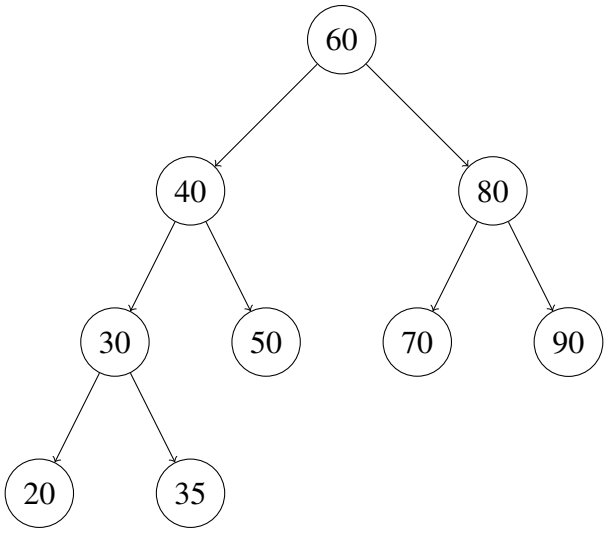


그림 3: Binary search tree

Answer:

그림 5 참조 .

(i) 그림 3의 트리에 10 을 추가하고, 모든 노드의 balance factor 가 0 혹은 ±1 이 되도록 조절한 결과를 그리시오.

Answer:

그림 6 참조

(j) 그림 3의 트리에서 50 을 삭제하고, 모든 노드의 balance factor 가 0 혹은 ±1 이 되도록 조절한 결과를 그리시오.

Answer:

그림 7 참조

4. Red-black tree 의 조건 다섯 가지를 기술하시오. 자료를 참조하거나 컴퓨터를 사용하여도 무방합니다.

Answer:

- Every node is either red or black
- The root is black
- Every leaf (including Nil) is black

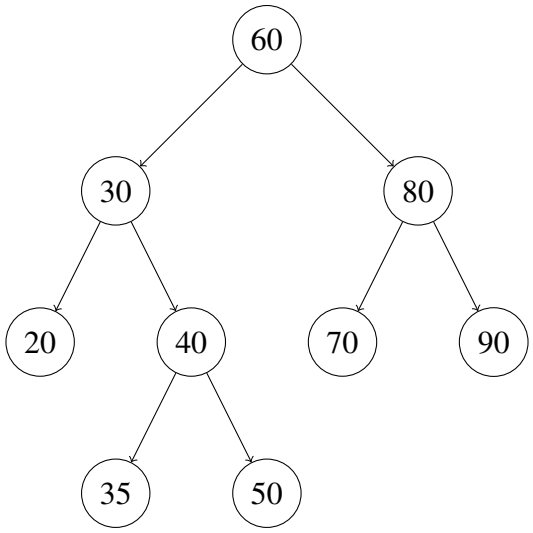


그림 4: Binary search tree (rotate right)

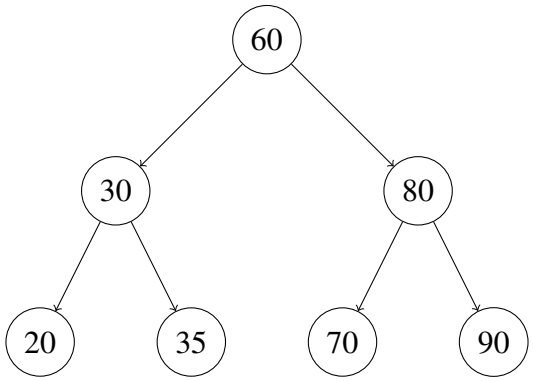


그림 5: Binary search tree (transplant)

- If a node is red then both its children are black
- For each node, all simple paths from the node to descendent leaves contain the same number of balck nodes

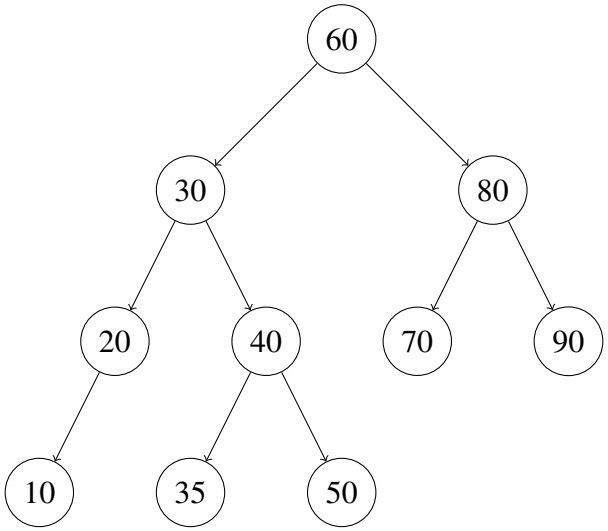


그림 6: Binary search tree (insert 10)

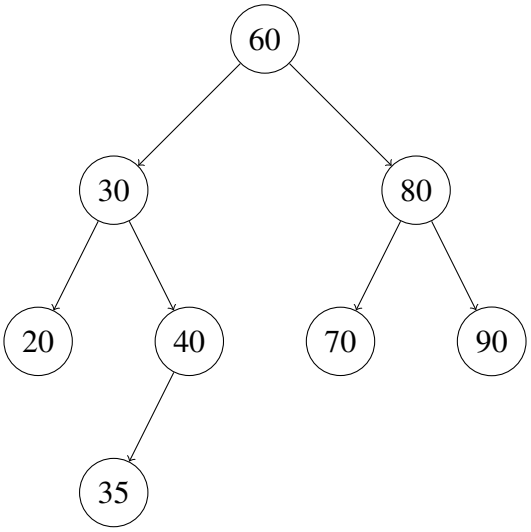


그림 7: Binary search tree (delete 50)