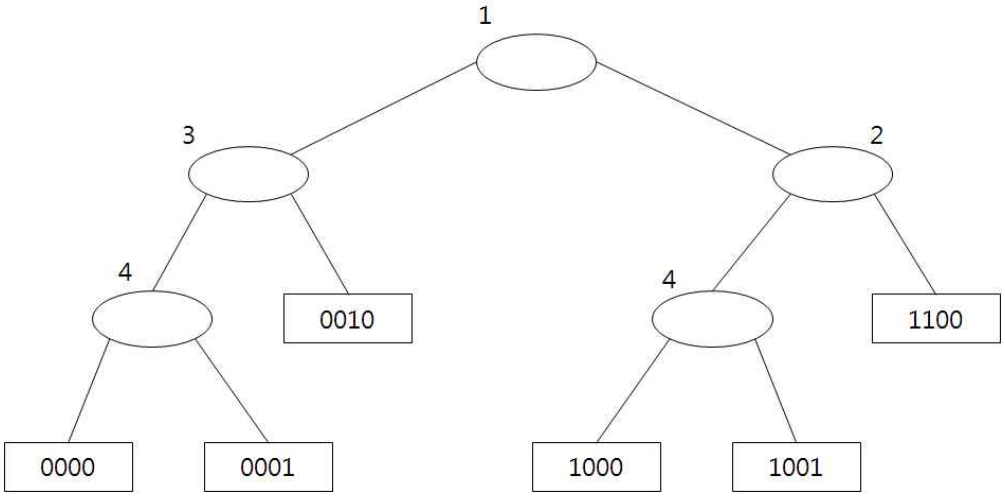
【 문제-1 】 (30점)

패트리샤에 관하여 다음 물음에 답하시오.

(1) 디지털 탐색 트리와 비교하여 패트리샤가 가지는 장점을 설명하시오. (6점)

(2) 압축 이진 트라이를 패트리샤로 변환하는 과정을 기술하고, 다음의 압축 이진 트라이를 패트리샤로 변환하시오. (단, 압축 이진 트라이의 노드 외부 번호는 이 노드에서 사용되는 키의 비트 번호 bitNumber를 나타낸다.) (8점)



(3) 패트리샤를 정의하기 위해 사용된 선언은 다음과 같다.

```
typedef struct patriciaTree *patricia;
   struct patriciaTree {
      int bitNumber;
      element data;
      patricia leftChild, rightChild;
      };
patricia root;
```

패트리샤 t를 탐색하는 함수 search(patricia t, unsigned k)를 C 언어로 완성하시오. (단, 이 함수는 탐색의 제일 마지막 노드에 대한 포인터를 반환하고, 만일 이 노드의 키가 k이면 탐색은 성공한다. i의 j번째 비트를 반환하는 함수 bit(i, j)를 이용하시오.) (8점)

```
patricia search(patricia t, unsigned k)
{
   patricia currentNode, nextNode;

   return nextNode;
}
```

(4) 공백 인스턴스로 시작하여 키 1000, 0010, 1001, 1100, 0000, 0001을 차례대로 패트리샤에 삽입하는 과정을 단계별로 나타내시오. (8점)

【 문제-2 】 (20점)

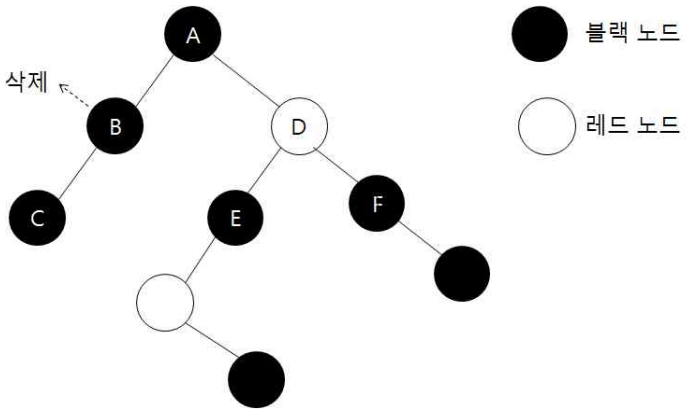
레드 블랙 트리의 블랙 노드를 삭제할 때에는 루트 노드와 모든 잎 노드 사이에 있는 블랙 노드의 수가 모두 동일해야 한다는 규칙이 깨진다.
RBT_RebuildAfterRemove()는 이 규칙을 유지하기 위해 회전을 포함하는 추가적인 처리를 하는 함수의 일부이다. RBTNode의 선언과 우회전 RBT_RotateRight() 함수는 아래와 같다. 좌회전 RBT_RotateLeft() 함수는 우회전 함수에서 사용하는 왼쪽 자식을 오른쪽 자식으로, 오른쪽 자식을 왼쪽 자식으로 바꿔 사용한다고 할 때, 다음 물음에 답하시오.

```
typedef struct tagRBTNode
{
    struct tagRBTNode* Parent;
    struct tagRBTNode* Left;
    struct tagRBTNode* Right;
    enum {RED, BLACK} Color;
    int Data;
} RBTNode;
```

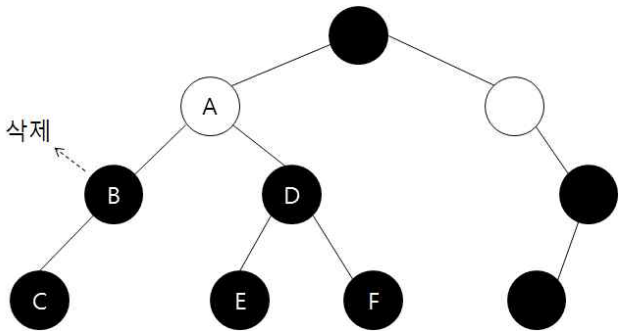
```
void RBT_RotateRight(RBTNode** Root, RBTNode* Parent)
{
    RBTNode* LeftChild = Parent->Left;
    Parent->Left = LeftChild->Right;
    if (LeftChild->Right != Nil)
       LeftChild->Right->Parent = Parent;
    LeftChild->Parent = Parent->Parent;
    if (Parent->Parent == NULL)
       (*Root)= LeftChild;
    else
    {
       if (Parent == Parent->Parent->Left)
          Parent->Parent->Left = LeftChild;
       else
          Parent->Parent->Right = LeftChild;
    }
    LeftChild->Right = Parent;
    Parent->Parent = LeftChild;
}
```

```
void RBT_RebuildAfterRemove(RBTNode** Root, RBTNode* Successor)
{
   RBTNode* Sibling = NULL;
   while (Successor->Parent != NULL && Successor->Color == BLACK)
   {
      if (Successor == Successor->Parent->Left)
      {
         Sibling = Successor->Parent->Right;
         if (Sibling->Color == RED)
         {
            _____(가-1)_____;
            _____(가-2)_____;
            RBT_RotateLeft(Root, Successor->Parent);
         }
         else
         {
            if (Sibling->Left->Color == BLACK &&
               Sibling->Right->Color == BLACK)
            {
               _____(나-1)_____;
               _____(나-2)_____;
            }
            else
            {
               if (Sibling->Left->Color == RED)
               {
                  Sibling->Left->Color = BLACK)
                  _____(다-1)_____;
                  RBT_RotateRight(Root, Sibling);
                  _____(다-2)_____;
               }
               Sibling->Color = Successor->Parent->Color;
               Successor->Parent->Color = BLACK;
               Sibling->Right->Color = BLACK;
               RBT_RotateLeft(Root, Successor->Parent);
               Successor = (*Root);
            }
         }
      }
      else
      {
         ...
      }
   }
    Successor->Color = BLACK;
}
```

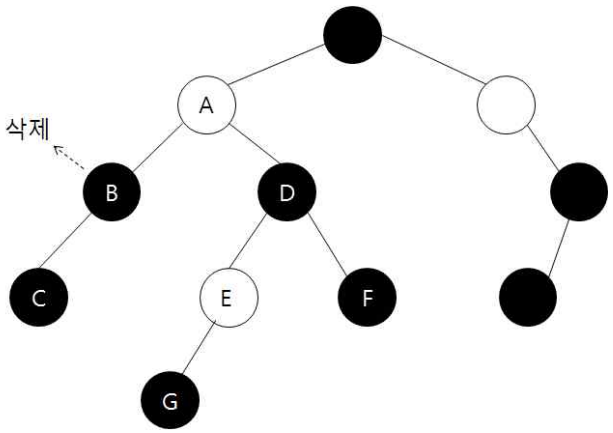
(1) 다음 그림과 같이 삭제되는 노드의 형제가 레드인 경우, (가-1), (가-2)에 해당하는 적절한 코드를 C 언어로 작성하시오. (6점)



(2) 다음 그림과 같이 삭제되는 노드의 형제가 블랙이고 형제의 양쪽 자식이 모두 블랙인 경우 (나-1), (나-2)에 해당하는 적절한 코드를 C 언어로 작성하시오. (7점)



(3) 다음 그림과 같이 삭제되는 노드의 형제가 블랙이고 형제의 왼쪽 자식이 레드, 오른쪽 자식이 블랙인 경우 (다-1), (다-2)에 해당하는 적절한 코드를 C 언어로 작성하시오. (7점)



【 문제-3 】 (30점)

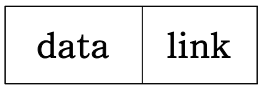
연결 리스트(Linked List)에 관하여 다음 물음에 답하시오.

(1) 단순 연결 리스트, 원형 연결 리스트, 이중 연결 리스트의 특징(개념)을 각각 설명하시오. (6점)

(2) 단순 연결 리스트의 다음 조건에서 첫 번째 노드를 삽입하는 알고리즘을 완성하시오. (8점)

(조건)

1) 노드의 논리적 구조는 다음과 같다.



2) 리스트 LST의 첫 번째 노드로 데이터 필드 값이 val인 i_node 노드를 삽입한다.
3) 치환, 대입, 연결은 ←로 표시한다.

(알고리즘)

```
insert_first_Node(LST, val)
    i_node ← getNode();
    
    
end insert_first_Node()
```

(3) 이중 연결 리스트의 다음 조건에서 중간 노드를 삽입하는 알고리즘을 완성하시오. (8점)

(조건)

1) 노드의 논리적 구조는 다음과 같다.

llink	data	rlink

2) 리스트 LST의 포인터 ago가 가리키는 노드 다음에 데이터 필드 값이 val인 i_node 노드를 삽입한다.

3) 치환, 대입, 연결은 ←로 표시한다.

(알고리즘)

```
insert_Node(LST, ago, val)
    i_node ← getNode();
    [                    ]
end insert_Node()
```

(4) 이중 연결 리스트의 다음 조건에서 중간 노드를 삭제하는 알고리즘을 완성하시오. (8점)

(조건)

1) 노드의 논리적 구조는 다음과 같다.

llink	data	rlink

2) 리스트 LST의 포인터 ers가 가리키는 노드를 삭제한다.

3) 치환, 대입, 연결은 ←로 표시한다.

(알고리즘)

```
delete_Node(LST, ers)
    [                    ]
end delete_Node()
```

【 문제-4 】 (20점)

정렬(Sort)에 관하여 다음 물음에 답하시오.

(1) 정렬 방법 가운데 기수 정렬(Radix Sort)의 처리 방법과 평균 시간 복잡도(Time Complexity)를 설명하시오. (4점)

(2) 주어진 데이터 값에 대한 선택 정렬(Selection Sort)을 수행하는 과정에서, 각 단계별로 수행이 종료된 값을 빈칸에 적고, 선택 정렬의 평균 시간 복잡도를 설명하시오. (단, 오름차순 정렬) (6점)

정렬 전	82	51	73	12	65	25	44	37
첫 번째 단계								
두 번째 단계								
세 번째 단계								
네 번째 단계								
다섯 번째 단계								
여섯 번째 단계								
일곱 번째 단계								
여덟 번째 단계								

(3) 주어진 데이터 값에 대한 버블 정렬(Bubble Sort)을 수행하는 과정에서, 각 단계별로 수행이 종료된 값을 빈칸에 적고, C언어로 구현한 버블 정렬 알고리즘을 중첩된 for문을 이용하여 완성하시오. (단, 오름차순 정렬) (10점)

정렬 전	82	51	73	12	65	25	44	37
첫 번째 단계								
두 번째 단계								
세 번째 단계								
네 번째 단계								
다섯 번째 단계								
여섯 번째 단계								
일곱 번째 단계								
여덟 번째 단계								

```
void bubble_sort(int i_data[], int n)
{
    int i, j, imsi;
    [                                        ]
}
```