자료구조론

문 1. 다음 탐색 알고리즘에 대한 설명으로 옳지 않은 것은?

```
int a_search(int list[], int key, int n) {
  int i;
  for(i=0; i<n; i++)
    if(list[i]==key) return i;
  return -1;
}
```

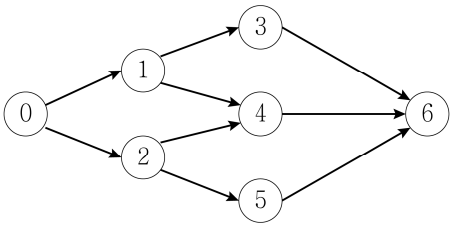
① 탐색에 성공하면 키 값의 인덱스를 반환한다.
② 최선의 경우, 시간 복잡도는 빅오 표기법으로 O(1)이다.
③ 최악의 경우, 시간 복잡도는 빅오 표기법으로 O(n)이다.
④ 평균적인 경우, 시간 복잡도는 빅오 표기법으로 O((n + 1)/2)이다.

문 2. 다음 C 코드의 실행 결과는?

```
#include <stdio.h>
int func (int n) {
  if(n <= 1) return 2;
  return func (n-1) + n;
}
int main (int argc, char* argv[]) {
  printf("%d\n", func (10) );
  return 0;
}
```

① 55 ② 56
③ 58 ④ 60

문 3. 다음 그래프에 대해 위상 정렬(topological sort) 알고리즘을 수행했을 때 생성되는 위상 순서 중 옳지 않은 것은?



① 0, 1, 2, 3, 4, 5, 6
② 0, 1, 3, 2, 4, 5, 6
③ 0, 2, 4, 5, 1, 3, 6
④ 0, 2, 5, 1, 3, 4, 6

문 4. 다음과 같이 배열을 이용하여 스택 자료구조를 정의하였다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은?

```
#define MAX 10
int stack[MAX];
int top = -1;
int push(int t) {
  if (top >= MAX - 1) {
    printf("\n Stack overflow.");
    return -1;
  }
  stack[____㉠____] = t;
  return t;
}
int pop(void) {
  if (top < 0) {
    printf("\n Stack underflow.");
    return -1;
  }
  return stack[____㉡____];
}
```

	㉠	㉡
①	++top	--top
②	++top	top--
③	top++	--top
④	top++	top--

문 5. 차수가 4이고 높이가 4인 m원 탐색 트리가 가질 수 있는 노드의 최대 수는? (단, 루트 노드만 있는 경우 트리의 높이는 1이다)

① 21
② 51
③ 85
④ 255

문 6. 다음은 이진트리를 전위 순회하여 얻어진 전위표기식이다. 이 트리에 대한 설명으로 옳지 않은 것은?

```
+ / * 6 2 - 3 1 * + 5 8 - 7 4
```

① 연산식의 최종 결과는 45이다.
② 트리로 재구성하면 루트 노드는 +이다.
③ 트리로 재구성하여 중위 순회하면 6*2/3−1+5+8*7−4이다.
④ 트리로 재구성하여 후위 순회하면 62*31/−58+74−*+이다.

문 7. 이진트리의 성질에 대한 설명으로 옳은 것은? (단, 루트 노드의 레벨은 0이고, 차수가 0인 노드의 개수는 n_0, 차수가 1인 노드의 개수는 n_1, 차수가 2인 노드의 개수는 n_2로 한다)

① 이진트리에서 단말노드의 개수를 n_0이라 할 때, n_2와의 관계식 $n_0 = n_2 + 1$을 만족한다.

② 깊이가 k인 이진트리가 가질 수 있는 노드의 최대 개수는 $2^{(k+1)}$이다($k \geq 0$).

③ 이진트리의 레벨 j에 있는 노드의 최대 개수는 $2^{(j-1)}$개이다 ($j \geq 0$).

④ 이진트리의 전체 노드 개수는 $n_0 + n_1 + n_2 - 1$개이다.

문 8. 다음 C 언어 코드를 실행한 후 5번째 및 7번째 줄에 출력되는 문장으로 옳은 것은?

```
#include <stdio.h>

void hanoi(int n, char from, char tmp, char to) {
  if( n==1 )
     printf("disk 1 from %c to %c\n", from, to);
  else {
     hanoi(n-1, from, to, tmp);
     printf("disk %d from %c to %c\n", n, from, to);
     hanoi(n-1, tmp, from, to);
  }
}

int main(int argc, char* argv[]) {
  hanoi(3, 'A', 'B', 'C');
}
```

	5번째 줄	7번째 줄
①	disk 1 from B to A	disk 1 from A to C
②	disk 1 from C to B	disk 2 from B to C
③	disk 2 from A to B	disk 3 from A to C
④	disk 3 from A to C	disk 2 from B to C

문 9. 다음과 같이 이중연결리스트를 이용하여 큐 자료구조를 정의하였다. add는 큐의 삽입 알고리즘이고, delete는 큐의 삭제 알고리즘이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은?

```
typedef struct _dnode {
  int key;
  struct _dnode *prev, *next;
} dnode;
dnode *head = (dnode*)malloc(sizeof(dnode));
dnode *tail = (dnode*)malloc(sizeof(dnode));
head->prev = head;   head->next = tail;
tail->prev = head;   tail->next = tail;

int add(int k) {
  dnode *t = (dnode*)malloc(sizeof(dnode));
  if (t == NULL) {
    printf("\n Out of memory.");
    return -1;
  }
  t->key = k;
          ㉠
  tail->prev = t;  t->next = tail;
  return k;
}

int delete(void) {
  dnode *t; int i;
  t = head->next;
  if (t == tail) {
    printf("\n Queue underflow.");
    return -1;
  }
  i = t->key;
          ㉡
  free(t);  return i;
}
```

① ㉠ tail->prev = t; t->prev = tail->prev;
㉡ head->next = t->next; t->next->prev = head;

② ㉠ tail->prev->next = t; t->prev = tail->prev->next;
㉡ head->next = t->next; t->next->prev = head;

③ ㉠ tail->prev->next = t; t->prev = tail->prev;
㉡ head->next = t->next; t->next = head;

④ ㉠ tail->prev->next = t; t->prev = tail->prev;
㉡ head->next = t->next; t->next->prev = head;

문 10. 다항식에서 하나의 항을 표현하기 위해 다음과 같은 구조체 poly를 정의하였다. 이 구조체를 이용하여 $A(x)=3x^{14}+2x^{8}+1$과 $B(x)=8x^{14}-4x^{9}+10x^{3}$을 저장하는 데 필요한 최소의 메모리 용량(byte)은? (단, 정수(int)를 표현하는 데 2 byte를 사용하고, 실수(float)를 표현하는 데 4 byte를 사용하며, 다항식의 모든 항은 poly 구조체를 이용하여 저장한다)

```
struct poly{
    int exp;
    float coef;
};
```

① 22 ② 24
③ 36 ④ 38

문 11. 다음은 행렬 연산을 하는 C 코드와 transpose함수 호출 후 행렬의 상태이다. ㉠에 들어갈 코드로 옳은 것은?

```
#include <stdio.h>
#define R 3
#define C 3
void transpose(int A[R][C], int B[R][C]) {
  for(int r = 0; r < R; r++)
   for(int c = 0; c < C; c++)
      ㉠
}
int main(int argc, char* argv[]) {
  int array1[R][C] = {{1,2,3},{4,5,6},{7,8,9}};
  int array2[R][C] = {0};
  transpose(array1, array2);
  return 0;
}
```

transpose함수 호출 후 행렬의 상태:

```
         1 2 3                    1 4 7
array1:  4 5 6          array2:   2 5 8
         7 8 9                    3 6 9
```

① A[c][r] = B[c][r];
② B[c][r] = A[c][r];
③ A[c][r] = B[r][c];
④ B[c][r] = A[r][c];

문 12. 열 우선(column major) 순서로 저장되는 3차원 배열 student[2][3][4]가 있을 때, 첫 번째 원소인 student[0][0][0]의 주소가 1024이면 student[1][2][2]의 주소는? (단, 배열 student의 각 원소의 크기는 24byte이고, 저장 순서는 [0][0][0],…,[0][2][3],[1][0][0],…,[1][2][3] 순서로 저장한다)

① 1168 ② 1192
③ 1504 ④ 1528

문 13. 다음 C 코드에서 함수 A와 B는 최대 공약수를 구하는 함수이다. 함수 B가 함수 A와 같은 결과를 출력하도록 ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은?

```
int A(int x, int y){
  int r;
  while (y != 0){
    r = x % y;  x = y;  y = r;
  }
  return x;
}

int B(int x, int y) {
  int r;
  if (_____㉠_____) return x;
  else {
    r = x % y;
    return _____㉡_____
  }
}
```

	㉠	㉡
①	y==0	B(y, r);
②	y==0	B(r, y);
③	y!=0	B(y, r);
④	y!=0	B(r, y);

문 14. C 언어에서 배열과 연결리스트에 대한 설명으로 옳지 않은 것은?

① 일반적으로 두 개의 정수형 데이터 세트 {1,2,3}과 {4,5,6,7,8}을 합병하여 {1,2,3,4,5,6,7,8}로 만드는 연산을 수행할 경우, 두 데이터 세트의 자료구조를 연결리스트로 만들면, 정수형 배열로 만드는 것보다 시간 복잡도를 낮게 구현할 수 있다.

② 일반적으로 정수형 데이터 세트 {1,2,3,4}를 {4,1,2,3}으로 변경하는 연산을 수행할 경우, 해당 데이터 세트의 자료구조를 연결리스트로 만들면, 정수형 배열로 만드는 것보다 더 빠르게 수행시킬 수 있다.

③ 일반적으로 정수형 데이터 세트 {1,2,3,4}를 {2,3,4,5}로 각 요소에 +1 연산을 수행할 경우, 데이터 세트의 자료구조를 연결리스트로 만들면, 정수형 배열로 만드는 것보다 메모리를 적게 차지한다.

④ 일반적으로 정수형 데이터 세트 {1,2,3,4}를 {0,1,2,3,4}로 새로운 값을 추가하는 연산을 수행할 경우, 데이터 세트의 자료구조를 연결리스트로 만들면, 정수형 배열로 만드는 것보다 더 빠르게 연산을 수행시킬 수 있다.

문 15. 다음은 기수정렬 알고리즘이다. ㉠이 두 번 실행된 후의 결과로 옳은 것은?

<조 건>

○ LSD(Least Significant Digit)는 가장 낮은 자릿수이고, MSD(Most Significant Digit)는 가장 높은 자릿수이다. 135의 경우 LSD 위치의 값은 5이고 MSD 위치의 값은 1이다.
○ 버킷은 10개 큐로 구현한다.
○ list는 배열로 구현하며 초깃값은 (437, 125, 85, 7, 632, 194)이고 n은 6이다.

```
RadixSort(list, n) {
  for(d ← LSD의 위치에서 MSD위치로 한 자리씩 이동){
    for(i ← list의 0번째 원소부터 n-1까지 반복){
      list의 i번째 원소를 d의 위치값에 따라 0번부터
      9번 사이의 버킷에 삽입한다.
    }
    for(b ← 0번 버킷부터 9번 버킷까지 반복){
      b번째 버킷에서 원소들을 순차적으로 읽어서 list에
      재배정한다.
    }
    ㉠list의 원소를 순서대로 출력한다.
  }
}
```

① 7 85 125 194 437 632
② 7 125 632 437 85 194
③ 632 194 125 85 437 7
④ 632 194 125 437 85 7

문 16. 데이터 <1, 3, 5, 7, 8, 10, 9>를 차례대로 하나씩 입력받아 이진 탐색 트리와 AVL 트리를 각각 만들었을 때, 두 트리 간의 높이 차는?

① 1
② 2
③ 3
④ 4

문 17. 초기 빈 상태의 트리에 다섯 개의 키 <1, 5, 3, 2, 6>를 차례대로 입력받아 디지털 탐색 트리를 만든 후, 전위 순회한 결과로 옳은 것은? (단, 각 키는 3비트 이진수로 표현한다)

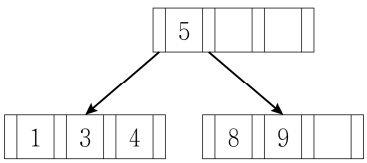
① 1, 3, 2, 5, 6
② 1, 3, 5, 2, 6
③ 1, 5, 2, 6, 3
④ 1, 5, 3, 2, 6

문 18. 입력받은 수식의 괄호쌍이 맞는지 스택 S를 이용하여 판단하는 알고리즘 DDD를 작성하였다. 수식에 존재하는 괄호는 소괄호 '()'이며, 다른 괄호는 수식에 존재하지 않는다. 수식의 길이는 n이며, 배열 X에 저장되어 입력된다. 알고리즘 DDD는 X를 입력받아 X에 저장된 수식의 괄호쌍이 맞으면 true, 그렇지 않으면 false를 반환하는 함수이다. ㉠~㉢에 들어갈 코드를 바르게 연결한 것은? (단, $S.push(c)$는 스택 S에 문자 c를 삽입하는 연산, $S.pop()$은 S에서 삭제하는 연산, $S.empty()$는 S가 비어 있으면 true, 아니면 false를 반환하는 함수이다)

```
Algorithm DDD(X, n)
Let S be an empty stack
for i←0 to n−1 do
  if X[i] = '(' then
    S.push(X[i])
  else if X[i] = ')' then
    if S.empty() then
      ㉠
    end if
    S.pop()
  end if
end for
if S.empty() then
  ㉡
else
  ㉢
end if
```

	㉠	㉡	㉢
①	return true	return true	return false
②	return false	return true	return false
③	return true	return false	return true
④	return false	return false	return true

문 19. 다음과 같은 2-3-4 트리에 키 6과 7을 차례로 삽입한 이후의 트리 상태에 대한 설명으로 옳은 것은?



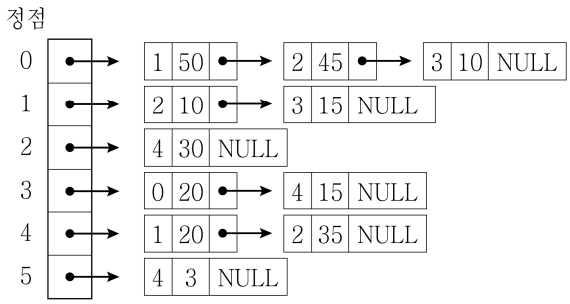
① 2-노드 1개, 3-노드 1개, 4-노드 2개로 구성된다.
② 루트 노드는 4-노드이다.
③ 가장 왼쪽의 리프 노드는 4-노드이다.
④ 가장 오른쪽의 리프 노드는 4-노드이다.

문 20. 다음 행렬은 그래프(graph)를 인접 행렬(adjacency matrix)로 나타낸 것이다. 이 그래프에서 솔린 알고리즘(Sollin algorithm)을 이용하여 최소 비용 신장 트리(minimum spanning tree)를 찾고자 한다. 알고리즘 수행 과정에서 나타나는 부분 그래프에 해당하지 않는 것은? (단, 부분 그래프는 간선의 집합으로 표현하며, 간선은 그래프에 포함된 두 정점 v, k에 대응하여 (v, k)의 형태로 표현한다)

정점	0	1	2	3	4	5	6
0	0	20	15	∞	∞	8	∞
1	20	0	5	12	∞	∞	∞
2	15	5	0	25	35	28	6
3	∞	12	25	0	7	∞	∞
4	∞	∞	35	7	0	∞	11
5	8	∞	28	∞	∞	0	9
6	∞	∞	6	∞	11	9	0

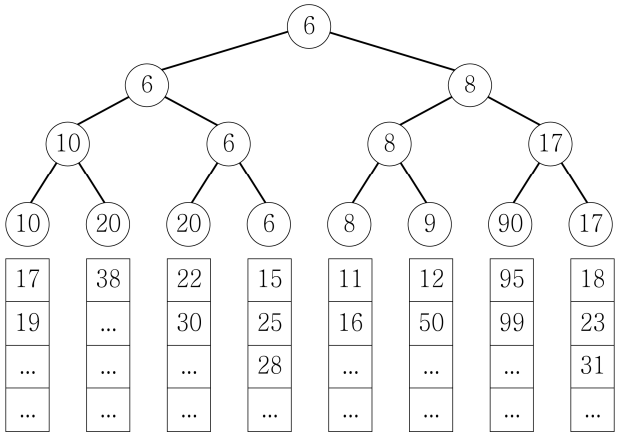
① { (0, 5) }
② { (1, 3), (3, 4) }
③ { (3, 4) }
④ { (1, 2), (2, 6) }

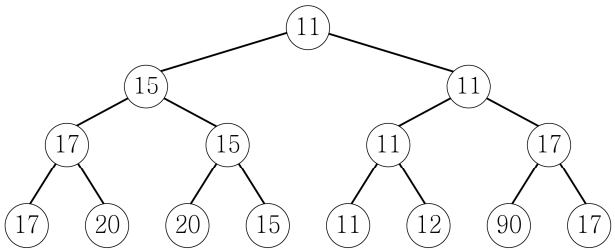
문 21. 다음은 방향그래프를 인접 리스트로 표현한 것이다. 이 그래프에 대한 설명으로 옳지 않은 것은?

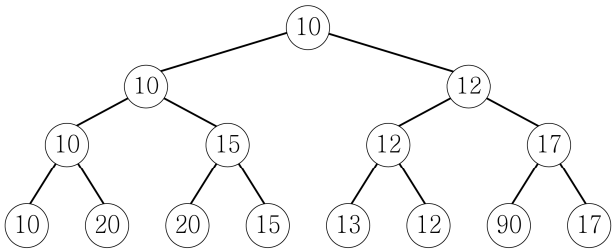


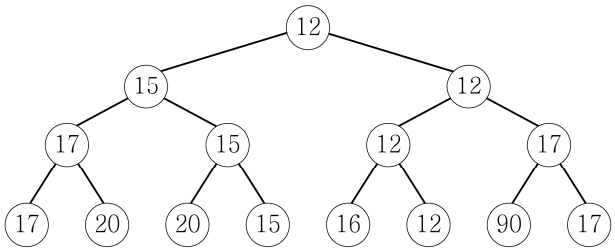
① 모든 사이클 경로는 (0,3,0), (2,4,2), (0,1,3,0), (1,2,4,1), (1,3,4,1), (0,2,4,1,3,0)이고, 그 경로에 각각 대응하는 길이는 2, 2, 3, 3, 3, 5이다.
② 각 정점(0,1,2,3,4,5)에 각각 대응되는 진입차수는 1, 2, 3, 2, 3, 0이다.
③ 각 정점(0,1,2,3,4,5)에 각각 대응되는 인접한 정점의 집합은 {1,2}, {2,3}, {4}, {0,4}, {1,2}, {4,2}이다.
④ 각 정점(0,1,2,3,4,5)에서 각각 대응되는 진출차수는 3, 2, 1, 2, 2, 1이다.

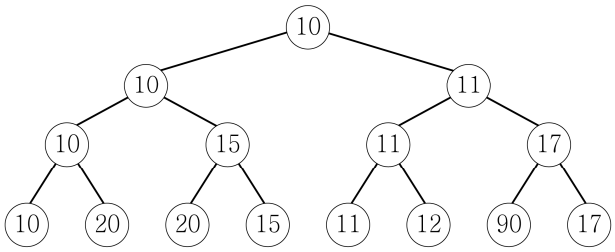
문 22. 승자 트리는 단말 노드(leaf node)에서부터 시작해서, 가장 작은 키 값을 갖는 노드(승자)를 찾아 부모로 올려서 루트 노드로부터 가장 작은 값을 찾아내는 방식이다. 승자 트리의 단말 노드에는 정렬된 배열(run)이 각각 달려 있고, 이 run에서 첫 번째 값을 읽어 가장 작은 값을 올려 내는 방식으로 모든 run을 비우면서 값을 정렬하게 된다. 8개의 run이 있는 아래 승자 트리에서 6, 8, 9 순서대로 3개의 레코드를 출력하고 난 뒤 run이 생략된 승자 트리의 형태로 옳은 것은?



①


②


③


④


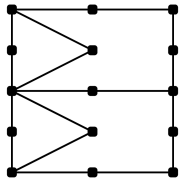
문 23. 다음 코드는 C 언어를 이용하여 주어진 연결 리스트 구조의 데이터에 대한 삽입 정렬 함수를 구현한 것이다. ㉠ ~ ㉤에 들어갈 코드를 바르게 연결한 것은?

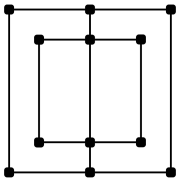
```
typedef struct _node {
  int data;
  struct _node *next;
}node;

void Insert(node **headRef, node *newNode) {
  if (*headRef == NULL ||
      (*headRef)->data >= newNode->data) {
    newNode->next = *headRef;
    *headRef = newNode;
  }
  else {
    node *current = *headRef;
    while (current->next != NULL &&
           current->next->data < newNode->data){
           ㉠
    }
    ㉡
    ㉢
  }
}

void Sort(node **headRef) {
  node *result = NULL, *current = *headRef;
  node *next;
  while (current!=NULL) {
    ㉣
    Insert(&result, current);
    ㉤
  }
  *headRef = result;
}
```

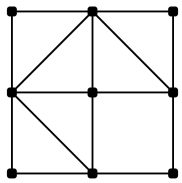
```
A : current->next = current;
B : current = current->next;
C : current->next = newNode;
D : next = current->next;
E : current->next = next;
F : current = next;
G : newNode->next = current->next;
```

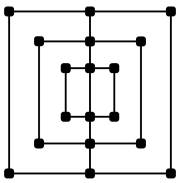
	㉠	㉡	㉢	㉣	㉤
①	A	D	E	F	G
②	B	G	C	D	F
③	C	E	B	A	D
④	E	G	B	F	A

문 24. 단순 무방향 그래프(simple undirected graph) G가 주어졌을 때, 정점(vertex)의 중복 방문을 허용하되, 간선(edge)의 중복은 허용하지 않는 경로(path)를 트레일(trail)이라고 한다. 그래프의 모든 간선들을 단 한 번씩만 통과하되 출발점과 도착점이 일치하지 않는 것을 오일러 트레일(Euler trail)이라고 할 때, 다음 그래프 중에서 오일러 트레일이 존재하지 않는 그래프는?

① 

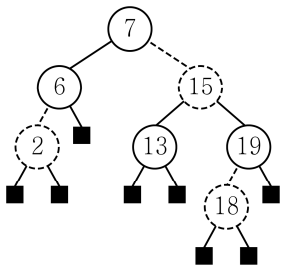
② 

③ 

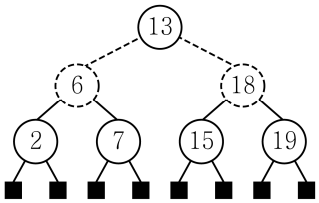
④ 

문 25. 비어 있는 레드-블랙(red-black) 트리에 <19, 18, 2, 15, 6, 13, 7>을 순서대로 입력한 결과 생성되는 레드-블랙 트리로 옳은 것은? (단, 점선 노드는 레드 노드, 점선 링크는 레드 링크이며, 실선 노드는 블랙 노드, 실선 링크는 블랙 링크이고, 리프 노드에 연결된 사각형의 단말은 외부 노드를 의미한다)

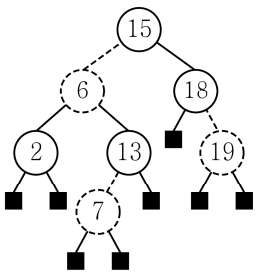
①



②



③



④