1. 주어진 자료에 맞는 정렬(sort) 알고리즘을 선택할 때 고려해야 할 사항으로 가장 옳지 않은 것은?
① 필요한 작업 공간 및 소요 시간
② 정렬할 자료의 양
③ 정렬에 필요한 메모리(memory)의 크기
④ 추가되는 데이터의 배열 상태

2. C언어로 작성된 〈보기〉의 함수는 길이가 n인 배열 array에 저장된 값들을 역순으로 재배치하는 프로그램이다. ㈎에 들어갈 내용으로 가장 옳은 것은?

〈보기〉
```
void reverse(int array[], int n)
{
        int t;
        for (int i = 0; i < n / 2; i++) {
                ㈎
        }
}
```

① t = array[i - 1];
array[i - 1] = array[n - i - 1];
array[n - i - 1] = t;
② t = array[i - 1];
array[i - 1] = array[n - i];
array[n - i] = t;
③ t = array[i];
array[i] = array[n - i - 1];
array[n - i - 1] = t;
④ t = array[i];
array[i] = array[n - i];
array[n - i] = t;

3. 비어 있는 최소 힙(min heap)에 〈보기〉에 주어진 데이터를 순서대로 하나씩 삽입 연산을 수행하여 모두 삽입한 후, 2번의 삭제 연산을 수행하였다. 최소 힙의 가장 마지막 레벨의 가장 왼쪽 노드에 저장된 데이터는? (단, 최소 힙은 완전 이진트리로 구현한다.)

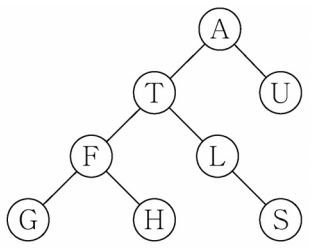
〈보기〉
153, 98, 180, 128, 105, 177, 192, 99, 116

① 128 ② 153 ③ 177 ④ 192

4. 후위표기식(postfix expression)으로 표현된 수식 중에 계산 결과가 나머지 수식과 다른 것은?
① A B C + *
② B C * A +
③ C A * B A * +
④ A B * A C * +

5. 〈보기 1〉과 같이 노드 A를 루트로 하는 이진트리를 〈보기 2〉의 규칙에 따라 구성할 때, 구성한 배열의 원소를 순서대로 바르게 나열한 것은? (단, 배열의 인덱스는 1부터 시작한다. 원소가 빈칸인 경우에는 '-'로 표시한다. 배열의 마지막이 연속적으로 빈칸인 경우 해당 부분을 생략하여 표시한다.)

〈보기 1〉



〈보기 2〉
㈎ 노드 i의 부모노드 인덱스 = i / 2 (정수부분)
㈏ 노드 i의 왼쪽 자식노드 인덱스 = 2 * i
㈐ 노드 i의 오른쪽 자식노드 인덱스 = 2 * i + 1

① A T U F L G H S
② A T F G U L H - S
③ A T U F L - - G H - S
④ A T U F - L - - - G H - - - - S

6. 버킷(bucket)당 슬롯(slot)이 1개이고, 모두 10개의 버킷으로 구성된 해시 테이블(hash table)에 대해서 해시함수(hash function) h가 〈보기 1〉과 같이 정의되며, 충돌(collision) 처리는 선형 조사법(linear probing)을 사용한다. 8개의 데이터를 해시 테이블에 삽입했을 때 해시 테이블의 모습이 〈보기 2〉와 같다면, 이와 같은 형태의 테이블이 나타날 수 없는 입력 순서는?

〈보기 1〉

$$h(k) = k \ mod \ 10$$

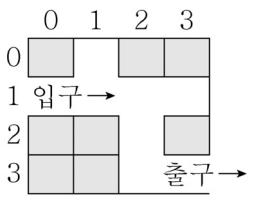
〈보기 2〉

[0]	80
[1]	61
[2]	30
[3]	
[4]	44
[5]	65
[6]	64
[7]	77
[8]	17
[9]	

① 77 65 44 61 80 30 17 64
② 61 44 80 65 77 17 64 30
③ 80 61 44 30 77 65 64 17
④ 44 64 65 80 61 30 77 17

7. 〈보기 1〉은 2차원 미로(maze)를 배열로 표현한 것이며, 〈보기 2〉는 이 구조를 이용하여 미로를 찾는 과정이다. 스택에 저장될 미로의 각 위치를 (행 번호, 열 번호)로 표현할 때, 〈보기 1〉에서 입구는 (1, 0), 출구는 (3, 3)이다. 〈보기 1〉의 미로에 대해서 〈보기 2〉의 미로찾기를 실행하는 과정 중에 스택에 나타날 수 없는 것은?

〈보기 1〉



〈보기 2〉

```
def maze_search()
        현재 위치를 입구로 초기화
        while(현재 위치가 출구가 아니면)
                (현재 위치를 방문한 것으로 표시)
                (현재 위치에서 이동이 가능한 위치 중에 아직
                방문하지 않은 위치들을 스택에 push)
                if(스택이 비어있으면)
                        return 실패
                else
                        스택에서 pop하여 현재 위치로 설정
        return 성공
```

① | (1, 1) |

② | (1, 3)
(1, 2)
(0, 1) |

③ | (1, 2)
(0, 1) |

④ | (3, 3)
(1, 3)
(0, 1) |

8. 〈보기〉의 인접 행렬(adjacency matrix)은 도시 사이의 경로값을 나타낸 것이다. ㈎ 서울에서 평택, ㈏ 서울에서 대구까지 각각 최단 경로값을 구하고자 한다. ㈎와 ㈏의 최단 경로값의 합은? (단, '∞'은 두 도시 간의 직접적인 연결이 없음을 의미하고, 출발지와 도착지가 같은 경로는 없다.)

〈보기〉

	서울	성남	평택	기흥	대구
서울	0	20	30	35	∞
성남	20	0	25	23	∞
평택	30	25	0	30	50
기흥	35	23	30	0	40
대구	∞	∞	50	40	0

① 75 ② 95 ③ 105 ④ 125

9. 관계식 $T(n)$이 〈보기〉와 같을 때, 옳지 않은 것은?

〈보기〉

$$T(n)=2T(n/2)+n,\ T(0)=T(1)=1$$

① $T(n)=\Omega(n^2)$ ② $T(n)=\Theta(n\log n)$

③ $T(n)=O(n^2)$ ④ $T(n)=O(n\log n)$

10. 〈보기〉는 정렬된 정수 배열 array에서 주어진 키 값(target)에 대한 이진 탐색(binary search)을 재귀(recursion) 방법을 통해 수행하는 함수 binarySearch()를 나타낸 것이다. ㈎와 ㈏에 들어갈 코드를 옳게 짝지은 것은?

〈보기〉

```
const int NOT_IN_ARRAY = -1;       // 배열에 없음
const int ARRAY_UNORDERED = -2; // 배열이 정렬되어 있지 않음
const int LIMITS_REVERSED = -3;   // 최대 색인이 최소 색인보다 큼

/* array[]: 입력 배열, lower: 이진 탐색 범위의 하단 색인, upper: 이진 탐색
범위의 상단 색인, target: 탐색하고자 하는 주어진 키 */
int binarySearch(int array[], int lower, int upper, int target) {
        int center, range;
        range = upper - lower;
        if ( range < 0 ) {
                return LIMITS_REVERSED;
        } else if( ( range == 0 ) && ( array[lower] != target ) ) {
                return NOT_IN_ARRAY;
        }
        if ( array[lower] > array[upper] )
                return ARRAY_UNORDERED;

        center = ( ( range ) / 2 ) + lower;
        if ( target == array[center] ) {
                return center;
        } else if ( target < array[center] ) {
                return _____(가)_____;
        } else {
                return _____(나)_____;
        }
}
```

① ㈎ binarySearch(array, lower − 1, center, target)
㈏ binarySearch(array, center − 1, upper + 1, target)

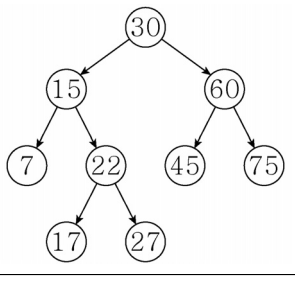
② ㈎ binarySearch(array, lower, center − 1, target)
㈏ binarySearch(array, center + 1, upper, target)

③ ㈎ binarySearch(array, center + 1, upper − 1, target)
㈏ binarySearch(array, lower − 1, center, target)

④ ㈎ binarySearch(array, center + 1, upper, target)
㈏ binarySearch(array, lower, center − 1, target)

11. 〈보기 1〉의 이진트리(binary tree) T에 대한 설명으로 옳은 것을 〈보기 2〉에서 모두 고른 것은?

〈보기 1〉



〈보기 2〉

ㄱ. 노드 22의 형제(sibling)는 노드 45이다.
ㄴ. T의 전위(preorder) 순회 결과는 노드 30 15 7 22 17 27 60 45 75 순이다.
ㄷ. T의 중위(inorder) 순회 결과는 노드 7 15 17 22 27 30 45 60 75 순이다.
ㄹ. T의 레벨 순서(level-order) 순회 결과는 노드 30 15 60 7 22 45 75 17 27 순이다.

① ㄱ, ㄴ, ㄷ ② ㄱ, ㄴ, ㄹ

③ ㄴ, ㄷ, ㄹ ④ ㄱ, ㄴ, ㄷ, ㄹ

12. 〈보기〉는 주어진 연결 리스트에서 끝에서부터 m번째에 위치한 요소(Element)를 반환하는 함수 findMToLastElement()를 C언어로 구현한 함수를 나타낸다. 이 함수에서 최종적으로 반환되는 mBehind 포인터는 끝에서부터 m번째에 위치한 요소를 가리킨다. (가)와 (나)에 들어갈 코드를 옳게 짝지은 것은?

〈보기〉

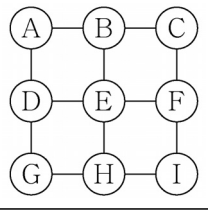
```
typedef struct Element{
        int data;
        struct Element* next;
} Element; //요소 구조체 정의

Element* findMToLastElement(Element *head, int m) {
/* head: 연결 리스트의 첫번째 요소를 가리키는 포인터, m: 연결
리스트의 m번째 요소 */
        Element *current, *mBehind;
        int i;
        current = head;
        mBehind = head;
        for (i = 0; i < m; i++) {
                if (current -> next) {
                        ____(가)____
                } else {
                        return NULL;
                }
        }
        while (current -> next) {
                ____(가)____
                ____(나)____
        }
        return mBehind;
}
```

	(가)	(나)
①	mBehind = mBehind -> next;	head = head -> next;
②	current = current -> next;	head = head -> next;
③	mBehind = mBehind -> next;	current = current -> next;
④	current = current -> next;	mBehind = mBehind -> next;

13. 〈보기〉에 주어진 무방향 그래프에서 깊이 우선 탐색 방식으로 방문할 수 있는 정점(vertex)들의 순서를 나열한 결과로 가장 옳지 않은 것은?

〈보기〉



① A-D-E-F-C-B-I-H-G
② A-D-G-H-E-B-C-F-I
③ A-D-G-H-E-F-I-B-C
④ E-D-G-H-I-F-C-B-A

14. 〈보기〉에서 함수 f( )의 시간복잡도는?

〈보기〉

```
int f(int n)
{
   int count = 0;
   for (int i = n; i > 0; i /= 2)
      for (int j = 0; j < i; j++)
         count += 1;
   return count;
}
```

① $\Theta(n)$ ② $\Theta(n\log n\log n)$
③ $\Theta(n\log n)$ ④ $\Theta(n^2)$

15. 8개의 정수를 퀵 정렬(quick sort) 알고리즘으로 정렬한다. 첫 번째 분할(partition) 연산을 수행한 결과가 〈보기〉와 같다고 할 때 설명으로 가장 옳은 것은?

〈보기〉

2 5 1 8 10 13 12 11

① 피봇으로 사용된 것은 8 또는 10이다.
② 피봇이 8일 수는 있지만, 10은 아니다.
③ 8은 피봇이 아니고, 10은 피봇이 될 수 있다.
④ 8과 10은 피봇이 아니다.

16. 비어있는 스택에 〈보기 1〉에 주어진 숫자들을 순서대로 〈보기 2〉의 규칙을 따라 처리할 때, 숫자들을 모두 처리한 후 스택에 저장된 숫자들을 모두 pop을 이용하여 출력한 결과로 가장 옳은 것은? (단, 〈보기 2〉의 top은 가장 최근에 스택에 push된 값을 가리킨다.)

〈보기 1〉

1, 5, 0, 0, −2, 6, 1, −2, 0, 10

〈보기 2〉

- 처리하려는 숫자를 K라고 하자.
- K가 0이 아닌 경우 아래의 조건 가운데 해당하는 항목을 수행한다.
  - 스택이 비어있거나 top이 가리키는 숫자가 K보다 큰 경우: K를 push 한다.
  - top이 가리키는 숫자가 K보다 작거나 같은 경우: pop 한 뒤 추출된 숫자에 K를 더한 값을 push 한다.
- K가 0인 경우 아래 조건 가운데 해당하는 항목을 수행한다.
  - 스택이 비어있는 경우: 다음 숫자를 처리한다.
  - 스택이 비어있지 않은 경우: pop한다.

① 10 ② 10 −2 1 4
③ 10 6 3 1 ④ 11 4

17. n개의 정점(vertex) V_0, …, V_{n-1}를 갖는 방향 그래프(directed graph)를 인접 행렬(adjacency matrix)로 표현할 때, 정점 V_x에서부터 V_y로의 간선(edge) (V_x, V_y)가 있으면 인접 행렬 a[x][y]에 1을 저장하고, 그렇지 않은 경우는 0을 저장한다. 정점 V_i의 진입차수(in-degree)와 진출차수(out-degree)를 계산하는 코드는?

①
```
in_degree = out_degree = 0;
for(j = 0; j <= n - 1; j++)
        in_degree = in_degree + a[ j ][ i ];
for(j = 0; j <= n - 1; j++)
        out_degree = out_degree + a[ i ][ j ];
```

②
```
in_degree = out_degree = 0;
for(j = 0; j <= n - 1; j++)
        in_degree = in_degree + a[ i ][ j ];
for(j = 0; j <= n - 1; j++)
        out_degree = out_degree + a[ j ][ i ];
```

③
```
in_degree = out_degree = 0;
for(i = 0; i <= n - 1; i++)
        in_degree = in_degree + a[ i ][ j ];
for(i = 0; i <= n - 1; i++)
        out_degree = out_degree + a[ j ][ i ];
```

④
```
in_degree = out_degree = 0;
for(i = 0; i++; i <= n - 1)
        in_degree = in_degree + a[ j ][ i ];
for(i = 0; i++; i <= n - 1)
        out_degree = out_degree + a[ i ][ j ];
```

18. 〈보기〉는 8개의 정수로 된 초기 입력 자료 40, 25, 71, 35, 17, 43, 95, 68을 오름차순으로 정렬한 결과이다. 이때 적용한 정렬 방법은?

〈보기〉

초기 입력 자료: 40 25 71 35 17 43 95 68

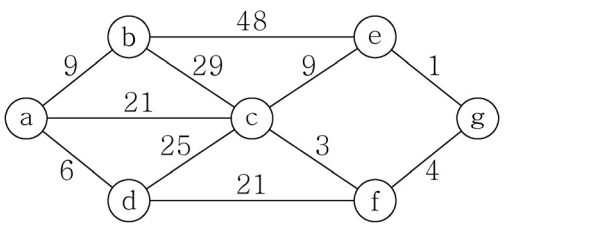
〈〈〈〈〈〈〈〈〈〈 정렬 수행 과정 〉〉〉〉〉〉〉〉〉〉

과정1 〉〉 25 40 71 35 17 43 95 68
과정2 〉〉 25 40 35 71 17 43 95 68
과정3 〉〉 25 35 40 71 17 43 95 68
과정4 〉〉 25 35 40 71 17 43 95 68
과정5 〉〉 25 35 40 71 17 43 68 95
과정6 〉〉 25 35 40 71 17 43 68 95
과정7 〉〉 17 25 35 40 43 68 71 95

① 삽입 정렬(insertion sort)
② 퀵 정렬(quick sort)
③ 버블 정렬(bubble sort)
④ 병합 정렬(merge sort)

19. 〈보기〉의 그래프에서 정점(vertex) a를 시작 노드로 하여 Prim 알고리즘을 적용해 최소 신장 트리(minimum spanning tree)를 생성할 때 포함되는 간선(edge)을 순서대로 바르게 나열한 것은?

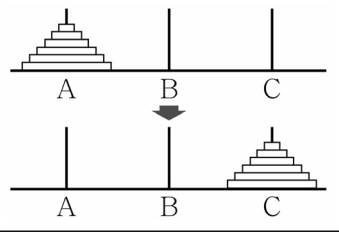
〈보기〉



① (a, d), (a, b), (d, f), (f, c), (f, g), (g, e)
② (a, b), (a, d), (d, f), (f, g), (g, e), (f, c)
③ (a, d), (a, b), (a, c), (c, f), (g, e), (f, g)
④ (e, g), (c, f), (f, g), (a, d), (a, b), (a, c)

20. 〈보기 1〉과 같이 세 개의 막대 A, B, C가 주어져 있고, 막대 A에는 N개의 크기가 서로 다른 원판이 큰 것부터 아래에 놓이도록 차례로 쌓여 있다. 한 막대의 맨 위에 있는 원판 하나를 꺼내 다른 막대의 맨 위에 놓을 수 있는데, 이때 작은 원판 위에 큰 원판을 올려놓을 수 없다. 막대 A에 있는 원판을 모두 막대 C로 옮기는 방법을 〈보기 2〉와 같이 재귀함수로 작성하고자 할 때, 〈보기 2〉의 (가)에 들어갈 내용은? (단, N은 3 이상이다. hanoi(N, start, to, via)는 막대 start의 맨 위에 있는 N개의 원판을 막대 to로 옮기는 함수이고, move(1, start, to)는 start의 맨 위 원판 1개를 to로 옮기는 함수이다.)

〈보기 1〉



〈보기 2〉

```
void hanoi(int N, int A, int C, int B){
    if(N == 1)
        move(1, A, C);
    else {
        (가)
    }
}
```

① hanoi(N-1, A, B, C); move(1, A, B); hanoi(N-1, B, C, A);
② hanoi(N-1, A, B, C); move(1, A, C); hanoi(N-1, B, C, A);
③ hanoi(N-1, A, C, B); move(1, A, B); hanoi(N-1, C, A, B);
④ move(1, A, B); hanoi(N-1, A, C, B); move(1, B, C);