

자료구조론

- 문 1. 다음의 D1 ~ D4에서 제시된 키(key) 값의 순서대로 공백(empty) 이진 탐색 트리(binary search tree)에 데이터를 삽입하여 T1 ~ T4를 만들었을 때, 모양이 다른 트리는? (단, 모양은 노드의 키 값과 간선의 연결 구조를 포함한다)

D1: 4, 2, 3, 6, 8, 5, 7 → 이진 탐색 트리 T1

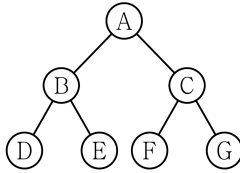
D2: 4, 2, 6, 8, 5, 3, 7 → 이진 탐색 트리 T2

D3: 4, 6, 3, 8, 2, 5, 7 → 이진 탐색 트리 T3

D4: 4, 6, 8, 5, 7, 2, 3 → 이진 탐색 트리 T4

- ① T1
② T2
③ T3
④ T4

- 문 2. 다음 최소 힙(min heap)에 대한 설명으로 옳지 않은 것은?



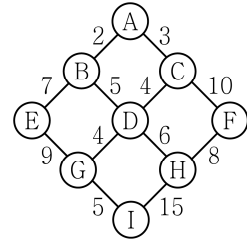
- ① 완전 이진 트리(complete binary tree)이다.
② 노드 A는 가장 작은 키 값을 가져야 한다.
③ 노드 B와 노드 C가 같은 키 값을 가질 수 있다.
④ 노드 C는 노드 D보다 작은 키 값을 가져야 한다.

- 문 3. 다음 인접 행렬(adjacency matrix)에 의해 표현되는 그래프에 대한 설명으로 옳은 것은? (단, 인접 행렬에서 $[i]$ 는 정점 i 를 의미한다)

	[0]	[1]	[2]	[3]	[4]
[0]	0	1	0	0	0
[1]	0	0	1	1	0
[2]	0	0	0	0	1
[3]	0	0	0	0	1
[4]	0	1	0	0	0

- ① 간선의 개수가 5개인 무방향 그래프(undirected graph)이다.
② 정점 0에서 정점 4까지의 단순 경로(simple path) 길이는 2이다.
③ 정점 3에 도달하기 위한 경로가 없는 정점이 있다.
④ 사이클(cycle)이 존재하는 방향 그래프(directed graph)이다.

- 문 4. 다음 가중 그래프(weighted graph)에 Kruskal 알고리즘을 적용하여 구성된 최소 비용 신장 트리(minimum cost spanning tree)의 최소 비용은?



- ① 36
② 39
③ 44
④ 46

- 문 5. 키 값이 1, 2, 3, 4, 5, 6, 7, 8인 8개의 노드로 구성된 이진 탐색 트리를 전위 순회(preorder traversal)하면, 키 값 6, 2, 1, 4, 3, 5, 8, 7 순서로 노드를 방문한다. 이 트리를 후위 순회(postorder traversal)할 경우 방문 노드의 키 값을 방문 순서대로 바르게 나열한 것은?

- ① 1, 2, 3, 4, 5, 6, 7, 8
② 1, 3, 2, 6, 5, 8, 7, 4
③ 1, 3, 5, 4, 2, 7, 8, 6
④ 8, 7, 6, 5, 4, 3, 2, 1

- 문 6. 차수(m)가 3인 공백 B-트리에 19, 18, 2, 15, 6, 13, 7을 순서대로 삽입하여 구성된 B-트리에 대한 설명으로 옳은 것은?

- ① 트리의 높이(height)가 1이다.
② 15는 루트(root) 노드에 저장되어 있다.
③ 3개의 자식(child) 노드를 가지는 노드가 존재한다.
④ 6이 저장된 노드와 18이 저장된 노드는 형제(sibling) 관계이다.

- 문 7. 스택을 이용하여 다음 후위 표현식(postfix expression)의 연산을 수행하였다. 최종 결과 값과 연산 과정에서 스택 내 피연산자의 최대 개수를 바르게 연결한 것은? (단, *은 곱셈 연산자이고 /은 나눗셈 연산자이다)

9 1 - 2 / 1 2 1 + * - 1 -

최종 결과 값 스택 내 피연산자의 최대 개수

- ① 0 3
② 0 4
③ 1 3
④ 1 4

- 문 8. 다음의 표는 단순 연결 리스트(singly linked list)를 표현한 것으로 각 행은 각 노드의 주소, 데이터 및 다음 노드 주소로 구성되어 있다. <다음 노드 주소 값>은 주소가 102인 노드를 삭제한 후 다음 노드 주소의 값을 설명한 것이다. ㉠ ~ ㉤에 들어갈 값을 바르게 연결한 것은? (단, 특정 노드의 다음 노드가 없을 때 그 노드의 다음 노드 주소 값은 NULL이다)

주소	데이터	다음 노드 주소
100	LEE	NULL
101	PARK	104
102	KIM	101
103	JUNG	102
104	SEO	100

— <다음 노드 주소 값> —

- 주소가 100인 노드의 다음 노드 주소는 (㉠)이다.
 ○ 주소가 101인 노드의 다음 노드 주소는 (㉡)이다.
 ○ 주소가 103인 노드의 다음 노드 주소는 (㉢)이다.
 ○ 주소가 104인 노드의 다음 노드 주소는 (㉤)이다.

㉠ ㉡ ㉢ ㉤

- ① NULL 104 101 100
 ② NULL 103 101 104
 ③ 101 103 102 104
 ④ 101 104 102 100

- 문 9. 다음 C 언어 프로그램의 출력 결과는? (단, 배열 h의 시작 주소는 6487296이고, 자료형 int와 float는 4바이트의 크기를 가진다)

```
#include <stdio.h>

int main()
{
    struct human {
        char sextype;
        union t {
            char children[3];
            char beard[3];
        } u;
        int age;
        struct date {
            int month;
            int day;
            int year;
        } dob;
        int personalID;
        float salary;
    } h[10];
    printf("%d, %d, %d", sizeof(struct human),
        &h[3].u.children[0] - &h[0].sextype,
        &h[5].u.beard[1] - &h[0].sextype);
}
```

- ① 28, 85, 142 ② 28, 86, 143
 ③ 31, 94, 157 ④ 31, 95, 158

- 문 10. 다음 C 언어 프로그램의 출력 결과는?

```
#include <stdio.h>

#define SWAP(x, y, t) ((t)=(x), (x)=(y), (y)=(t))

void recursive(char *list, int i, int n) {
    int j, temp;
    if (i == n) {
        for (j = 0 ; j <= n ; j++)
            printf("%c", list[j]);
        printf(" ");
    }
    else {
        for (j = i ; j <= n ; j++) {
            SWAP(list[i], list[j], temp);
            recursive(list, i + 1, n);
            SWAP(list[i], list[j], temp);
        }
    }
}

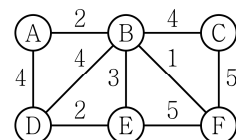
int main()
{
    char list[3] = {'a', 'b', 'c'};
    recursive(list, 0, 2);
}
```

- ① ab ac bc
 ② ab bc ca
 ③ abc acb bac bca cba cab
 ④ abc acb bca bac cab cba

- 문 11. 행 우선(row major) 순서로 저장되는 C 언어 2차원 배열 A[5][5]에서 다른 값을 갖는 것은? (단, 임의의 배열 인덱스(index) $i(0 \leq i \leq 4)$ 와 $j(0 \leq j \leq 4)$ 에 대해 A[i][j]가 $(i - j)$ 의 값을 가진다)

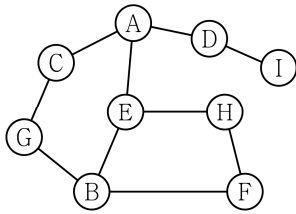
- ① A[3][1]
 ② $*(A + 3) + 1$
 ③ $*(A[3] + 1)$
 ④ $**((A + 3) + 1)$

- 문 12. 다음 가중치 그래프에 Prim 알고리즘을 적용하여 최소 비용 신장 트리(minimum cost spanning tree)를 만들 때, 최소 비용과 세 번째로 선택된 간선의 가중치의 합은? (단, A를 시작 정점으로 한다)



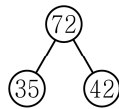
- ① 13
 ② 14
 ③ 15
 ④ 16

문 13. 다음 그래프에 대한 너비 우선 탐색(BFS: Breadth First Search)의 방문 순서는? (단, 시작 정점은 A이고, 인접한 정점들은 알파벳 순서로 방문한다)



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

문 14. 다음에 주어진 최대 힙(max heap)에 키 값이 2, 14, 49, 3, 30, 41, 27, 17인 8개의 데이터를 순서대로 삽입하여 최대 힙을 구성하였다. 구성된 최대 힙에서 모든 리프(leaf) 노드의 키 값의 합은?



- ① 63
- ② 73
- ③ 108
- ④ 113

문 15. <조건>을 만족하고 front와 rear의 값이 모두 0인 원형 큐(circular queue)에서 <동작>에 나열된 연산을 순서대로 모두 수행하였다. 연산 완료 후 원형 큐에 존재하는 모든 유효 데이터와 front 및 rear의 값을 바르게 연결한 것은? (단, enqueue(x)에 의해 삽입된 후 dequeue()에 의해 삭제되지 않은 데이터만 유효 데이터로 인정한다)

＜조 건＞

- 원형 큐는 원소 개수가 6개인 1차원 배열로 구현되었다.
- front와 rear의 값이 같을 때 원형 큐는 공백 상태이다.
- front와 $((rear + 1) \bmod 6)$ 의 값이 같을 때 원형 큐는 포화 상태이다.
- enqueue(x)에 의해 rear의 값을 $((rear + 1) \bmod 6)$ 로 변경한 후 rear가 가리키는 위치에 데이터 x 를 삽입한다. 단, 포화 상태일 때에는 데이터 삽입이 이루어지지 않아 원형 큐의 상태가 변하지 않는다.
- dequeue()에 의해 front의 값을 $((front + 1) \bmod 6)$ 로 변경한 후 front가 가리키는 위치에서 데이터를 빼내어 삭제한다. 단, 공백 상태일 때에는 데이터 삭제가 이루어지지 않아 원형 큐의 상태가 변하지 않는다.

＜동 작＞

```
enqueue(2) → enqueue(5) → enqueue(7) → enqueue(10) →
dequeue() → enqueue(11) → enqueue(15) → enqueue(17) →
dequeue() → dequeue() → enqueue(20)
```

	<u>모든 유효 데이터</u>	<u>front</u>	<u>rear</u>
①	10, 11, 15, 17, 20	3	1
②	10, 11, 15, 20	3	1
③	10, 11, 15, 17, 20	4	2
④	10, 11, 15, 20	4	2

문 16. 키 값 1, 2, 3, 4, 5를 가지는 5개의 레코드로 이루어진 리스트를 퀵 정렬(quick sort)로 오름차순 정렬하고자 한다. 정렬 과정에서 총 비교 횟수가 가장 많은 리스트는? (단, 피벗(pivot)은 정렬하고자 하는 대상의 첫 번째 레코드로 선택한다)

- ① 1, 2, 3, 4, 5
- ② 2, 3, 1, 4, 5
- ③ 3, 1, 2, 4, 5
- ④ 3, 1, 2, 5, 4

문 17. 다음의 키 값을 가지는 7개의 데이터를 순서대로 삽입하여 AVL 트리를 구성한 후 후위 순회를 수행하였다. 방문 노드들의 키 값을 방문 순서대로 바르게 나열한 것은?

10, 4, 7, 12, 3, 13, 15

- ① 3, 4, 10, 15, 13, 12, 7
- ② 3, 7, 4, 12, 15, 13, 10
- ③ 3, 7, 4, 15, 13, 12, 10
- ④ 3, 10, 13, 15, 4, 12, 7

문 18. 다음 C 언어 프로그램의 출력 결과는?

```
#include <stdio.h>
```

```
void func(char s[], int size) {
    if (size <= 0)        return;
    s[size-1] = s[size-1] + 1;
    func(s,size-2);
}
```

```
int main()
{
    char str[] = "window";
    func(str, 6);
    printf("%c", str[3]);
}
```

- ① d ② e
③ n ④ o

문 19. 다음은 1차원 배열 arr에서 두 번째로 큰 수를 찾는 C 언어 함수이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? (단, 배열 arr의 모든 원소는 서로 다른 양의 정수이고, n은 배열의 크기이며 2보다 크거나 같다)

```
void Largest2nd(int *arr, int n){
    int i, max1 = 0, max2 = 0;

    for (i = 0; i < n; i++){
        if (____⑦____){
            max2 = max1;
            max1 = arr[i];
        }
        else if (____⑧____){
            max2 = arr[i];
        }
    }

    printf("Second Largest Number = %d\n", max2);
}
```

- | $\textcircled{T}$ | $\textcircled{L}$ |
|---------------------------------|--|
| ① $\text{arr}[i] > \text{max1}$ | $\text{arr}[i] > \text{max2} \ \&\& \ \text{arr}[i] < \text{max1}$ |
| ② $\text{arr}[i] > \text{max1}$ | $\text{arr}[i] < \text{max2} \ \&\& \ \text{arr}[i] > \text{max1}$ |
| ③ $\text{arr}[i] < \text{max1}$ | $\text{arr}[i] > \text{max2} \ \&\& \ \text{arr}[i] < \text{max1}$ |
| ④ $\text{arr}[i] < \text{max1}$ | $\text{arr}[i] < \text{max2} \ \&\& \ \text{arr}[i] > \text{max1}$ |

문 20. 다음은 원형 연결 리스트(circular linked list)의 맨 앞에 insert_node가 지정하는 노드를 삽입하는 C 언어 함수이다. ㉠, ㉡에 들어갈 내용을 바르게 연결한 것은? (단, last_ptr은 마지막 노드를 지정하는 포인터 변수를 가리키는 주소 값을 가진다)

```
struct node {
    int data;
    struct node *link;
};
typedef struct node *node_ptr;

void InsertFront(node_ptr *last_ptr,
                 node_ptr insert_node) {
    if (*last_ptr == NULL) {
        *last_ptr = insert_node;
        ㉠
    }
    else {
        insert_node->link = (*last_ptr)->link;
        ㉡
    }
}
```

㉠

㉡

- ① insert_node = insert_node->link; (*last_ptr) = insert_node;
 ② insert_node = insert_node->link; (*last_ptr)->link = insert_node;
 ③ insert_node->link = insert_node; (*last_ptr) = insert_node;
 ④ insert_node->link = insert_node; (*last_ptr)->link = insert_node;