

16. 다음 설명 중 옳은 것은?

- ① 컴퓨터는 주기억장치와 보조기억장치를 이용해 자료를 처리하고 저장한다.
- ② 자료(Data)는 정보(Information)를 처리(Process)해서 얻어진 결과(Result)다.
- ③ 알고리즘은 프로그램의 표현을 통해 추상화된다.
- ④ 자료구조는 자료의 저장과 이용을 위한 추상화이다.

17. 명령어를 실행하는 주체가 누구이든 똑같은 결과를 생성해야 하며, 똑같이 실행 가능한 명령어를 전제로 하는 알고리즘의 조건은 무엇인가?

- ① 출력 ② 유효성
③ 입력 ④ 명확성

18. 알고리즘의 성능 측정이나 예측에 대한 것이 아닌 것은?

- ① 알고리즘 실행 시간의 예측
- ② 실행 메모리의 예측
- ③ 실행 시간의 측정
- ④ 실행 메모리의 측정

19. 다음의 그림이 의미하는 저장방식은 무엇인가?

A[0][0]	A[0][1]	A[0][2]	A[0][m-1]
A[1][0]
A[2][0]
...
A[n-1][0]	A[n-1][m-1]

- ① 행우선 저장 ② 열우선 저장
③ 배열우선 저장 ④ 리스트우선 저장

20. 다음 행렬 A 를 '희소행렬 표현 행렬 B' '으로 구현할 경우에 $B[0,1]$ 의 값은 무엇인가?

$$A = \begin{bmatrix} 0 & 20 & 0 & 0 & 9 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 78 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 67 & 0 & 0 & 0 & 0 \\ 0 & 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 91 & 0 & 0 & 44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19 & 0 & 0 & 27 & 0 \end{bmatrix}$$

- ① 9 ② 8
③ 20 ④ 1

21. '1차원 배열 A'에서 첫 번째 원소 A[0]의 메모리 시작주소를 '100'이라고 하고, A[]의 크기를 '5' 라고 가정한다. 다음 중 A[3]의 저장 주소는 무엇인가?

- [illegible]

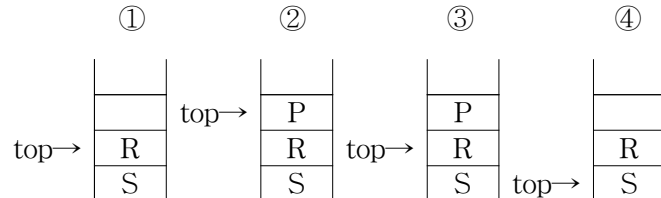
22. 다음 스택의 연산에서 [가]와 [나]에 맞는 것은 무엇인가?

```
Stack Push(stack, item) ::=
    if ([가])
        then { [나] }
        else { 스택의 가장 위에 item을 삽입하고, 스택을
반환한다; }
```

- ① IsFull(stack), 'stackFull'을 출력한다;
- ② IsEmpty(stack), 'StackEmpty'을 출력한다;
- ③ IsFull(stack), 'stackEmpty'을 출력한다;
- ④ IsEmpty(stack), 'stackFull'을 출력한다;

23. 다음 연산들을 수행할 경우에 ⑥번 연산의 결과는 무엇인가?

- ① CreateS(3);
- ② Push(stack, 'S');
- ③ Push(stack, 'T');
- ④ Pop(stack);
- ⑤ Push(stack, 'R');
- ⑥ Push(stack, 'P');
- ⑦ Push(stack, 'Q');
- ⑧ Pop(stack);



24. 다음은 배열을 이용해 스택을 구현하고 스택에 데이터를 삽입하는 과정을 나타내는 코드이다. [가]에 들어갈 코드로 가장 알맞은 것은 무엇인가?

<스택의 생성>

```
#define STACK_SIZE 10
typedef int element;
element stack[STACK_SIZE];
int top = -1;
```

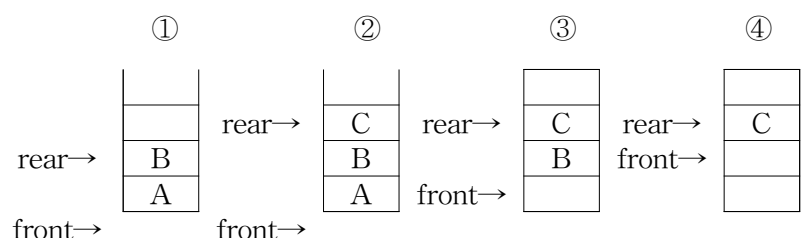
<삽입 과정>

```
void push(element item) {
    if ( isFull() ) {
        printf( "Stack is Full!! \n" );
        return;
    }
    else
        stack[++top] = item;
}
```

- ① top == -1 ② top <= STACK_SIZE
③ top >= STACK_SIZE ④ top >= STACK_SIZE - 1

25. 다음 연산들을 수행할 경우에 ④번 연산의 수행 결과는 무엇인가?

- ① Create_q(4);
- ② Add_q(queue, 'A');
- ③ Add_q(queue, 'B');
- ④ Add_q(queue, 'C');
- ⑤ Delete_q(queue);
- ⑥ Delete_q(queue);
- ⑦ Delete_q(queue);
- ⑧ Add_q(queue, 'D');



26. 다음 설명 중 틀린 것은 무엇인가?
- ① 큐에서는 원소의 삭제연산이 이루어지는 곳을 앞(front)이라 한다.

② 큐 생성 함수(Create_q(maxQueueSize))를 호출하기만 하면 프로그래머가 지정한 크기의 새로운 큐를 생성할 수 있다.

③ Boolean IsFull_q(queue, maxQueueSize) 연산은 큐가 가득 찼는지를 확인한다.

④ Queue Add_q(queue, item) 연산은 큐에 새로운 원소를 삽입하고 front를 하나 증가시킨다.

27. 다음은 배열을 이용해 큐를 구현하고 큐에 데이터를 삭제하는 과정을 나타내는 코드이다. [가]에 들어갈 코드로 가장 알맞은 것은 무엇인가?

```
<큐의 생성>
#define QUEUE_SIZE 5
typedef int element;
element queue[QUEUE_SIZE];
int front = -1; // 삭제가 발생하는 곳
int rear = -1; // 삽입이 발생하는 곳

<삭제 과정>
element Delete_q(int *front, int rear) {
    if (*front == rear) {
        printf("Queue is empty \n");
        return; }
    return ( [가] );
}
```

- ① queue[++(*front)]
- ② queue[--(*front)]
- ③ queue[--(*rear)]
- ④ queue[(*rear)++]

28. 리스트에 대한 설명으로 틀린 것은 무엇인가?
- ① 물품이나 사람의 이름 따위를 일정한 순서로 적어 놓은 것이다.

② 논리적 ‘순서’가 메모리 공간에서의 물리적인 위치를 순서적으로 결정한다.

③ 원소들 간의 ‘의미적인 순서’의 의미를 갖는다.

④ 포인터를 이용하면 효율적으로 구현할 수 있다.

29. 다음 프로그램의 실행 결과는 무엇인가?

```
float a, *p_a;
float b, *p_b;
p_a = (float *)malloc(sizeof(float));
p_b = (float *)malloc(sizeof(float));
*p_a = 33.9;
*p_b = 33.1;
*p_a = *p_b;
*p_b = 33.9;
printf("a is %f, b is %f\n", *p_a, *p_b);
free(p_a);
free(p_b);
```

- ① a is 33.1, b is 33.9
- ② a is 33.9, b is 33.1
- ③ a is 33.1, b is 33.1
- ④ a is 33.9, b is 33.9

30. 다음은 포인터로 구현된 단순 연결 리스트에서 마지막 노드 뒤에 새로운 노드를 삽입하는 연산을 나타낸 것이다. [가]에 들어갈 가장 알맞은 코드는 무엇인가? (단, x의 값은 100이라고 가정한다.)

```
void addNode(linkedList_h* H, int x) {
    listNode* NewNode
    listNode* LastNode
    NewNode = (listNode*)malloc(sizeof(listNode));
    NewNode → data = x;
    NewNode → link = NULL;

    if ( H → head == NULL) {
        H → head = NewNode
        return;
    }

    LastNode = H → head;
    while(LastNode → link != NULL)
        [가]
    LastNode → link = NewNode;
}
```

- ① LastNode = LastNode → link;
- ② LastNode = NewNode → link;
- ③ NewNode = LastNode → link;
- ④ NewNode = NewNode → link;