문제

1. 순서 리스트(ordered list) $(a_1, a_2, \cdots, a_n)$는 1차원 배열 또는 단순 연결리스트(singly linked list)를 이용하여 구현할 수 있다. 1차원 배열의 경우가 단순 연결리스트의 경우와 비교하여 상대적인 장점이 아닌 것은? (다만, 1 ≤ i ≤ n이다.)

① i번째 자료의 탐색이 빠르다.
② 새로운 자료의 삽입이 용이하다.
③ 고정된 수의 자료를 저장할 경우 공간 이용이 효율적이다.
④ i번째 자료의 수정이 빠르다.
⑤ 순서 리스트를 역순으로 탐색할 때 용이하다.

2. 데이터 구조에 대한 설명으로 올바른 것은?

① 배열구조는 삽입과 삭제가 빈번하게 일어나는 작업에 적합하다.
② 배열 크기는 아무리 커지더라도 접근(Access)속도는 일정하다.
③ 연결구조는 원소의 삽입과 삭제 시 이동(repacking) 시간이 많이 필요하다.
④ 연결구조는 외부단편화(fragmentation)로 인해 전체 기억공간을 효율적으로 사용할 수 없다.

3. 자료구조에 대한 설명이다. 잘못된 항목은?

① 원형 연결리스트(circular linked list)에서 head 포인터가 마지막 노드를 가리키게 하면 마지막 노드 다음에 삽입하는 것이 매우 간단해진다.
② 스택이나 큐는 배열 또는 연결리스트로 표현 가능하나 트리와 그래프는 배열 또는 연결리스트로 표현 불가능하다.
③ 원형 연결리스트(circular linked list)는 head 포인터가 마지막 노드를 가리키게 하면 첫 번째 노드 앞에 삽입하는 것이 매우 간단해진다.
④ 이중원형연결리스트는 원형 리스트가 리스트를 뒤로 순회할 수 없다는 점과 삭제하고자 하는 노드에 대한 포인터만으로는 그 노드를 삭제할 수 없다는 단점을 보완했다.

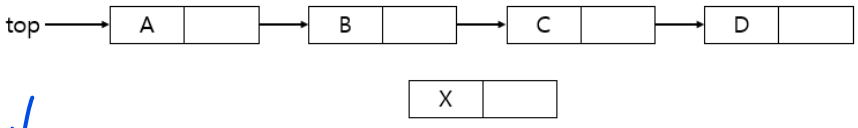
4. 배열에 관한 설명으로 가장 적합하지 않은 것은?

① 정렬된 배열에서 임의의 원소를 찾기 위해서는 순차 탐색을 사용하는 것이 바람직하다.
② 배열은 인덱스와 값의 쌍으로 구성된 집합이다.
③ 배열의 각 원소는 동일한 자료형을 갖는다.
④ 배열의 원소 값은 인덱스 값을 알면 쉽게 찾을 수 있다.
⑤ 배열은 순서 리스트(ordered list)를 표현하기 위하여 사용할 수 있다.

5. 연결 리스트(linked list)에 대한 설명으로 옳지 않은 것은?

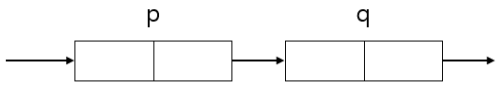
① 단순 연결리스트(singly linked list)에서는 한쪽 방향으로만 이동이 가능하다.
② 연결리스트(linked list)는 원형(circular)이 아닐 수도 있다.
③ 일반적으로 이중 연결리스트의 노드는 적어도 3개의 필드를 가진다.
④ 이중 연결리스트는 단순 연결리스트보다 모든 면에서 우수하다.
⑤ 연결리스트는 헤드 노드(head node)를 갖도록 구성할 수 있다.

6. 연결리스트의 삽입 알고리즘 순서 (B 노드 다음에 X 노드 삽입 시)는?



① Link(X)=Link(B)
Link(B)=X
② Link(B)=X
Link(X)=Link(B)
③ Link(X)=B
Link(B)=X
④ Link(B)=Link(X)
Link(X)=Link(B)
⑤ Link(X)=B
Link(B)=Link(X)

7. 다음 그림과 같은 단순 연결리스트에서 q를 삭제하는 직업을 C언어로 바르게 표현한 것은?



① p → link = q → link; free(p);
② p → link = q → link; free(q);
③ q → link = p → link; free(p);
④ q → link = p → link; free(q);

8. 다음은 연결리스트의 길이를 구하는 알고리즘이다. ( )안에 알맞은 것은?

```
int length(list_pointer head) {
    int count;
    list_pointer pt;
    if(head) {
        pt = head;
        do {
            ++count;
            (              );
        }while(pt != head);
    }
    return count;
}
```

① head = head→next
② pt = pt→next
③ pt = head→next
④ head = pt→next

9. 단순 연결 원형 리스트(singly linked circular list)에 대한 설명으로 가장 적절하지 않은 것은?

① 단순 연결리스트(singly linked list)의 단점을 보완한 것이다.
② 포인터 연결 방향의 반대 방향으로 직접 이동할 수 없다.
③ 삭제하고자 하는 노드에 대한 포인터만으로 그 노드를 삭제할 수 없다.
④ 단점을 보완하기 위해 이중 연결 원형 리스트(doubly linked circular list)를 사용할 수 있다.
⑤ 마지막 노드의 링크 필드 값은 널(null)이다.

10. 이중 연결 리스트(doubly linked list)에 대한 설명으로 옳지 않은 것은?

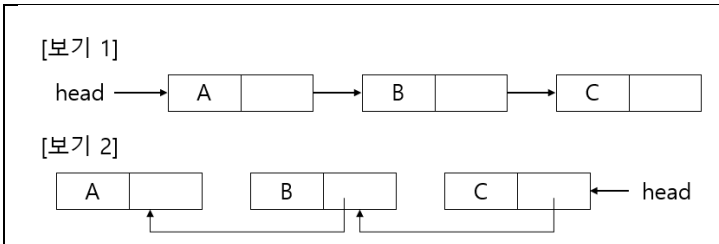
① 단순 연결리스트의 노드에 비하여 링크 필드가 더 필요하다.
② 특정 노드의 왼쪽과 오른쪽에 새로운 노드의 삽입과 삭제가 쉽다.
③ 원형으로 만들 경우 어떤 노드의 포인터 필드 값 1개를 잃어버려도 복구할 수 있다.
④ 같은 리스트를 단순 연결리스트에 저장했을 때보다 시작 노드로부터 n 번째 노드를 찾는 시간이 더 빠르다.
⑤ 임의의 노드에 대한 포인터 값이 그 노드의 왼쪽 노드와 오른쪽 노드에 저장된다.

11. 이중연결리스트 구조에서 노드 X 앞에 노드 Y를 추가할 경우, 4단계 중에서 3번째 단계 ㉢에 들어갈 내용은?

㉠ Y.Link <- X.Link ㉡ Y.Rlink <- X ㉢ ( ) ㉣ X.Link <- Y

① (X.Linlk)Rlink<-Y
② Rlink(Y) <- X
③ Llink(X)<-Rlink(Y)
④ (Y.Llink)Rlink <- X

12. 보기 1의 리스트를 보기2와 같은 리스트로 변경하는 알고리즘이다. (㉠), (㉡)에 알맞은 것을 고르면?




```
list_pointer middle = Null, trail; {
    while(lead != Null) {
        trail = middle;
        middle = lead;
        lead = lead→link;
        ( ㉠ )
    }
    ( ㉡ )
}
```

① ㉠ middle→link = trail ㉡ return middle
② ㉠ trail→link = trail ㉡ return trail
③ ㉠ lead→link = trail ㉡ return lead
④ ㉠ middle→link = trail ㉡ return trail
⑤ ㉠ trail→link = trail ㉡ return lead

13. 다음은 연결리스트에 대한 설명이다. 틀린 것은?

① 자료를 중간에 삽입할 때 한 자리씩 이동이 발생한다.
② 기억공간을 효율적으로 활용할 수 있다.
③ 연결된 원소들은 기억장소 내에 아무 곳에나 위치할 수 있다.
④ 큐나 스택을 연결리스트로 표현할 수 있다.

14. 다음 중 연결리스트의 장점은?

① 중간에 새로운 원소를 삽입 또는 삭제할 때 가장 효율적으로 처리된다.
② 기억장소의 낭비가 전혀 없는 자료 구조이다.
③ 원하는 자료의 액세스 시간이 가장 빠른 구조이다.
④ 자료를 정렬(Sort)하거나 탐색(Search)하기에 가장 좋은 구조이다.

15. 다음은 Singly Linked List에 대한 설명이다. 이들 중 틀리게 설명된 것은?

① list의 각 노드를 위치에 관계없이 저장한다. ② 각 노드는 link 부분을 갖고 있으며, 다음 노드의 주소를 갖는다. ③ 마지막의 link 부분은 맨 처음 node의 주소를 갖는다.

① 1 ② 2 ③ 3 ④ 1, 2

16. 다음은 원형 연결리스트에 대한 설명이다. 잘못된 것은?

① 무한 루프에 빠질 수 있다.
② 처음이 아닌 현재 노드부터 선행 노드를 검색할 수 있다.
③ 역방향 추적이 가능하다.
④ 검색을 종료할 수 있는 노드가 필요하다.

17. 스택에 관한 설명으로 옳지 않은 것은?

① 자료의 입력과 출력은 한 쪽 끝으로 이루어진다.
② LIFO(Last-In-First-Out)이라고도 한다.
③ 프로그램 언어에서 서브루틴의 호출에 사용
④ 운영체제의 작업 스케줄링에 사용

18. 스택 S 가 배열 Stackarray[1,...,Max]로 표현되어 있다. 아래 프로그램 중 스택 S 에 원소 X 를 삽입하는 프로시저로서 가장 옳은 것은? (단, 초기조건은 Top=0 이다.)

①
```
If(Top>Max) Then
    Stack_Overflow;
    Else Begin
        Top : = Top + 1;
        Stackarray[Top] : = X;
    End;
```

②
```
If(Top>Max) Then
    Stack_Overflow;
    Else Begin
        Stackarray[Top] : = X;
        Top : = Top + 1;
End;
```

③
```
If(Top>=Max) Then
    Stack_Overflow;
    Else Begin
        Top : = Top + 1;
        Stackarray[Top] : = X;
    End;
```

④
```
If(Top>=Max) Then
    Stack_Overflow;
    Else Begin
        Stackarray[Top] : = X;
        Top : = Top + 1;
End;
```

19. 스택에서 삽입 연산을 push, 삭제하여 출력하는 연산을 pop이라 할 경우 순서화된 원소 A, B,

push, push, pop, push, push, pop, pop, pop

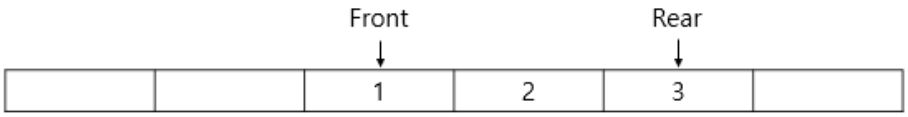
C, D에 대해서 다음 순서로 스택 연산을 수행하는 경우 출력되는 결과는?

① A B D C
② A B C D
③ B D A C
④ B A C D
⑤ B D C A

20. 산술식 A*(B+C)/D를 후위 표기법으로 표현하면?

① AB+*CD/
② ABC+*D/
③ ABCD/+*
④ ABC+D*/

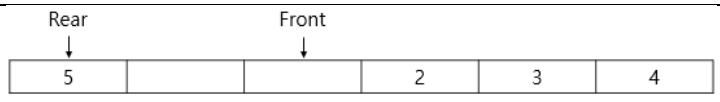
21. 다음 그림은 원형큐의 초기 상태를 나타낸 것이다. 초기 상태에서 ①②③과 같은 삽입, 삭제 연산을 했을 때 큐의 상태로 올바른 것은?

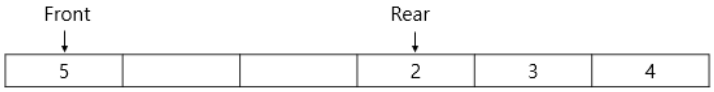


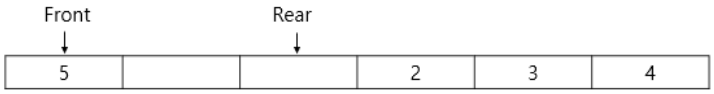
① insert(4)

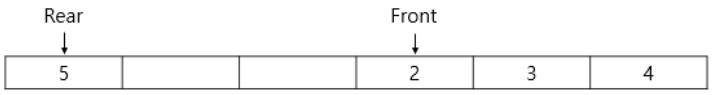
② delete()

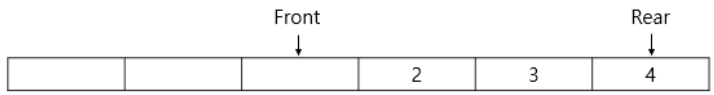
③ insert(5)

① 

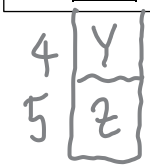
② 

③ 

④ 

⑤ 

22. 큐의 현재 상태는 다음과 같다. 이를 원형큐로 운영할 때 다음의 명령 수행 결과 Head와 Tail 값은 어떻게 되는가?



① Head = 1, Tail = 1

② Head = 3, Tail = 4

③ Head = 3, Tail = 2

④ Head = 5, Tail = 1