데이터 구조 중간고사 (2025. 4. 24.) 학번: 이름:

※ 시험 시간은 45분이며 뒷면에도 문제가 있습니다.

1. 다음에 선언된 변수(data)를 위해 할당된 메모리 공간을 바이트 단위로 적으시오. (각 3점, 단 32비트 운영체제와 컴파일러를 가정)

1) char data[5][10]; 50

2) char *data[5]; 20

3) double *data[5]; 20

4) 4

```
typedef struct ListNode {
    int num;
    struct ListNode *link;
} ListNode;
ListNode *data;
```

5) 8

```
typedef struct ListNode {
    int num;
    struct ListNode *link;
} ListNode;
ListNode data;
```

2. 다음의 각 식 또는 코드의 시간 복잡도를 빅오표기법으로 나타내시오 (각 3점, 단 n은 데이터의 개수를 의미함).

1) $n^2+2^n+2n!+4n$ O(n!)

2) $4n^3+11n^2+36nlog_2n+17n$ O(n^3)

3) O(n^2)

```
for (i=0; i<n; i++)
    for (j=i; j<n; j++)
        printf("%d  ", i*j);
```

4) 문제오류 -> 모두 정답

```
for (i=0; i<n; i++)
    for (j=0; j<n; j*=2)
        printf("%d  ", i*j);
```

5) O(log_2n)

```
int func(int list[], int low, int high, int key)
{
    int middle = (low + high)/2;
    if (low > high)
        return -1;
    else {
        if (list[middle] == key)
            return middle;
        else {

            if (list[middle] > key) {
                high = middle - 1;
                func(list, low, high, key);
            }
            else {
                low = middle + 1;
                func(list, low, high, key);
            }
        }
    }
}
```

위 함수 func(list, 0, n-1, key)로 호출한 경우, 시간복잡도는?

3. 각 문제에서 시간복잡도를 증명하시오 (각 5점)

1) $f(n)=2n^2+3n+4$이 $O(n^2)$임을 증명하시오.

빅오 표기법: f(n), g(n)에 대해 모든 $n>n_0$에 대해 $|f(n)| \le c|g(n)|$을 만족하는 2개의 상수 c와 n_0가 존재하면 $f(n)=O(g(n))$이 성립한다.

$n_0=1, c=5$에 대해 $2n^2+3n+4 \le 5n^2$이 성립함

2) $f(n)=2n^2+3n+4$이 $\Theta(n^2)$임을 증명하시오.

빅세타 표기법: f(n), g(n)에 대해 모든 $n>n_0$에 대해 $c_1|g(n)| \le |f(n)| \le c_2|g(n)|$을 만족하는 3개의 상수 c_1, c_2와 n_0가 존재하면 $f(n)=\Theta(g(n))$이 성립한다.

$n_0=1, c_1=2, c_2=5$에 대해 $2n^2 \le 2n^2+3n+4 \le 5n^2$이 성립함

4. 다음 코드들이 수행된 후, 화면 출력값을 그대로 쓰시오(각 5점).

1) func(1234) 수행 시 출력 결과: 1 2 3 4

```
void func(int n)
{
    if (n == 0)
        return;
    func(n/10);
    printf("%d ", n%10);
}
```

2) func(8, 8) 수행 시 출력 결과: 1 2 4 8

```
void func(int n, int i)
{
    if (i <= 0)
        return;
    func(n, i-1);
    if (n % i == 0)
        printf("%d ", i);
    return;
}
```

3) printf("%d", func(6)); 출력 결과: 11

```
int func(int x)

{
    if (x == 1) return 1;
    return (func(x/2) + func((x+1)/2) + 1);
}
```

5. 다음 문제에서 요구하는 표기식으로 변환하시오(각 5점).

1) 다음 중위표기식을 후위표기식으로 변환하시오.

5+4*3/(6-2)%7 543*62-/7%+

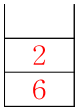
2) 다음 중위표기식을 전위표기식으로 변환하시오.

5/2*3-(1+4)*2 -*/523*+142

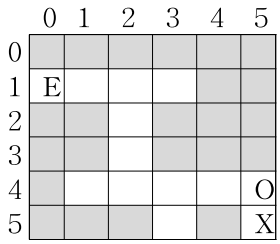
6. 다음 각 문제에서 스택의 상황을 정확히 표현하시오. (각 5점)

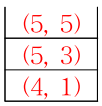
1) 다음 후위표기식 계산 함수에서 '+' 연산자를 읽어 처리하기 직전에 스택에 들어 있는 피연산자(또는 피연산자들)를 스택 그림으로 나타내시오.

후위표기식: 42/3*75-+1-



2) 다음의 미로 그림은 스택을 이용한 생쥐의 미로 찾기 알고리즘 동작에서 출발점 'E'부터 시작하여 목적지인 'X'에 도착하는 미로 찾기를 수행하고 있는 과정을 보여준다. 현재 생쥐는 'O'로 표시된 좌표 (4, 5)에 도착한 후, 향후 방문할 좌표 push 동작을 모두 수행한 시점이다. 이 시점에서 스택 안에 들어 있는 좌표(들)를 스택 그림으로 그려라. (단, 이 알고리즘에서 스택에 방문할 좌표 push는 위, 아래, 왼쪽, 오른쪽 순서로 수행된다.)





7. 다음 각 문제에서 요구하는 코드를 기입하여 프로그램을 완성하여라. (각 5점)

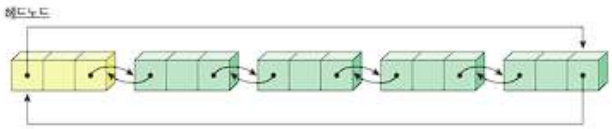
1) 다음에 선언된 형식의 노드들로 구성되는 단순 연결리스트에서 현재 head로 포인팅되는 연결리스트의 노드 개수를 리턴하는 node_count 함수를 완성하시오.

```
typedef int element;
typedef struct ListNode {
        element data;
        struct ListNode *link;
} ListNode;

int node_count(ListNode *head)
{
    int count=0;
    ListNode *p=NULL;

    for (p=head; p; p=p->link)
        count++;

    return count;
}
```

2) 앞의 문제 1)에서 선언한 형식의 단순 연결리스트에서 리스트를 역순(노드 연결 순서를 반대로)으로 구성하는 reverse 함수를 완성하시오.

```
//  역순으로 구성한 리스트의 헤드를 리턴함
ListNode *reverse(ListNode *head)
{
    ListNode *p, *q, *r;

    p = head;
    q = NULL;
    while (p)
    {
        r = q;
        q = p;
        p = p->link;
        q->link = r;
    }
    return q;
}
```

3) 아래 그림과 같이 구성되는 이중 연결리스트를 사용하며, 노드들의 형식은 아래 코드와 같다. (head 노드는 데이터를 가지지 않음)




```
typedef int element;
typedef struct DListNode {
        element data;
        struct DListNode* llink;
        struct DListNode* rlink;
} DListNode;
```

새로운 data를 가지는 노드를 before로 포인팅되는 노드 다음에 삽입하는 dinsert() 함수를 완성하시오.

```
void dinsert(DListNode *before, element data)
{
    DListNode *p = (DListNode *)malloc(sizeof(DListNode));
    p->data= data;

    p->data= data;
    p->llink = before;
    p->rlink = before->rlink;
    before->rlink->llink = p;
    before->rlink = p;

    return;
}
```

4) 3)과 같은 이중 연결리스트에서 인수 removed로 포인팅되는 노드를 삭제하는 ddelete() 함수를 완성하시오.

```
void ddelete(DListNode* head, DListNode* removed)
{
    if (removed == head) return;

    removed->llink->rlink = removed->rlink;
    removed->rlink->llink = removed->llink;

    free(removed);
}
```

5) 3)과 같은 이중 연결리스트에서 인수 key와 동일한 데이터를 가지는 노드의 포인터를 리턴하는 dsearch() 함수를 완성하시오. (key와 동일한 값을 가진 여러 개의 노드가 존재하더라도

첫 번째 노드 포인터만 리턴하면 되고, 없으면 NULL 리턴)

```
DListNode *dsearch(DListNode *head, element key)
{
    DListNode* p=NULL;

    for (p=head->rlink; p != head; p=p->rlink) {
          if (p->data == key)
               return p;
    }
    return NULL;

}
```