(1) 10진수 -49.6875을 32비트 단정도 부동 소수점 형식으로 나타낸 것은?

① 1 10000100 10001101000000000000000
② 1 10000100 10001101100000000000000
③ 1 10000101 10001101000000000000000
④ 1 10000101 10001101100000000000000

정 답 : ②

정답해설

32비트 단정도 부동 소수점은 부호 1비트, 지수부 8비트, 가수부 23비트로 이루어져있다.
부호비트에는 양수는 0을, 음수는 1을 표시해야 하므로 -49.6875의 부호비트는 1이다.
다음으로 -49.6875의 정수부인 49를 이진수로 표현하면 110001, 소수부인 0.6875를 이진수로 표현하면 0.1011이므로 연결하면 110001.1011이 된다. 여기서 정수부가 1이 되도록 소수점을 이동하면 1.100011011×2^5 이 된다. 여기서 가수부에는 정수부를 생략하고 소수부(100011011)만 저장한다. 마지막으로 지수부는 정규화해서 구한 지수와 바이어스를 더한 값을 저장해야 하므로 $5 + 127 = 132 = 10000100_{(2)}$ 을 저장하면 된다.
따라서 답은 ② 1 10000100 10001101100000000000000

출제근거 도서명	부동소수점을 이해하고 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	34, 35

(2) 함수 $f(n)=2n^2-n+12$ 이고, 상수 c 값이 3 라고 할 때 이 함수의 성능을 빅-오 표기법으로 하고, 식을 만족하는 n_0 의 최솟값을 구하시오

① 3
② 4
③ 5
④ 6

정 답 : ①

정답해설

빅-오 표기법이란 함수 $f(n)$ 과 $g(n)$ 이 주어졌을 때, 모든 $n \geq n_0$ 에 대하여 $|f(n)| \leq c|g(n)|$ 을 만족하는 상수 c 와 n_0 이 존재하면. $f(n) \leq O(g(n))$ 이다.

따라서 문제에서 주어진 식을 대입하면 $2n^2-n+12 \leq 3n^2$ 이 되므로 주어진 방정식을 계산하면 $n \geq 3$ 이 된다. 따라서 답은 3 이고 빅-오 표기법으로 나타내면 $f(n)=O(n^2)$ 이다.

출제근거 도서명	빅-오 표기법을 이해하고 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	53

(3) 다음은 행렬의 어떤 연산을 계산하는 프로그램이다. S[2][4]의 모든 원소의 합을 구하시오.

```c
# include <stdio.h>

void multiply() {
    int A[2][3] = { {2, 3, -5}, {4, -3, 6} };
    int B[3][4] = { {7,4,3,-2}, {-5,1,0,2}, {2,3,-1,4} };
    int S[2][4];

    int i, j, k;
    int mul;

    for (i = 0; i < 2; i++)
        for (j = 0; i < 4; j++) {
            mul = 0;
            for (k = 0; k < 3; k++) {
                mul += A[i][k] * B[k][j];
            }
            S[i][j] = mul;
        }
}
```

① 65
② 70
③ 75
④ 80

정 답 : ④

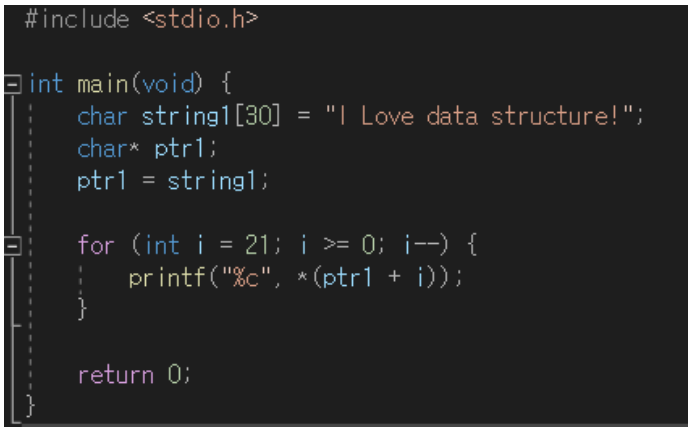
정답해설

위 프로그램은 두 행렬의 곱을 계산하는 행렬이다. A와 B에 각각 행렬을 초기화하고 두 행렬의 곱을 S에 저장한다. 계산한 값은 다음과 같다. $S=\begin{bmatrix}-11 & -4 & 11 & -18\\ 55 & 31 & 6 & 10\end{bmatrix}$

따라서 S의 모든 원소의 합을 구하면 $-11+(-4)+11+(-18)+55+31+6+10=80$ 이 된다.

출제근거 도서명	2차원 배열의 개념을 이해하고 c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	78

(4) 다음은 포인터를 이용해 문자열을 처리하는 프로그램이다. 다섯 번째로 출력되는 문자를 구하시오.



```c
#include <stdio.h>

int main(void) {
    char string1[30] = "I Love data structure!";
    char* ptr1;
    ptr1 = string1;

    for (int i = 21; i >= 0; i--) {
        printf("%c", *(ptr1 + i));
    }

    return 0;
}
```

① c
② u
③ r
④ t

정 답 : ④

정답해설

위 프로그램은 i 의 값이 21부터 하나씩 줄어들며 0이 될 때 까지 *(ptr+i)를 출력한다. 즉 초기화된 문자열을 거꾸로 출력하는 프로그램이다. 따라서 !erutcurts atad evoL I 이 출력되므로 다섯 번째로 출력되는 문자는 ④ t 이다.

출제근거 도서명	포인터를 이해하고 포인터를 이용해 문자열을 처리하는 등 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	89

(5) 다음은 구조체를 이용해 두 점의 내적을 구하는 프로그램이다. 이 프로그램의 출력 값을 구하시오.

```c
# include <stdio.h>

struct point {
    int xpos;
    int ypos;
};

int main(void) {
    struct point pos1 = { 2,3 };
    struct point pos2 = { 5,6 };
    struct point* pptr = &pos1;

    (*pptr).xpos += 4;

    pptr = &pos2;
    pptr->ypos -= 5;

    int inner = (pos1.xpos * pos2.xpos) + (pos1.ypos * pos2.ypos);
    printf("두 점의 내적은 %d입니다", inner);
    return 0;

}
```

① 18
② 20
③ 28
④ 33

정 답 : ④

정답해설

위 프로그램을 보면 구조체를 통해 xpos, ypos를 선언하고, 메인함수에서 pos1를 {2, 3}, pos2를 {5, 6}으로 초기화했다. *pptr을 pos1를 참조하므로 (*pptr).xpos를 통해 pos1의 xpos에 4 더해진다는 것을 알 수 있다. 즉 pos1은 {6, 3}이 된다. 또한 pptr이 pos2를 다시 참조하고 pptr -> ypos를 통해 pos2의 ypos에 5가 빼진다는 것을 알 수 있다. 즉 pos2는 {5, 1}이 된다. 마지막으로 inner는 (pos1.xpos * pos2.xpos) + (pos1.ypos + pos2.ypos) 로 계산할 수 있으므로 $(6\times5)+(3\times1)=33$이 된다. 따라서 답은 ④ 33

출제근거 도서명	구조체를 이해하고 점 연산자와 화살표 연산자를 사용해 이를 활용할 수 있다. 윤성우의 열혈 C프로그래밍, 윤성우, 오렌지미디어, 2010	페이지	464

(6) 다음은 제곱의 값을 계산하는 재귀함수이다. power(3, 5) 함수는 몇 번 실행되는가?

```c
# include <stdio.h>
# pragma warning (disable : 4996)

int main(void) {
    int x, n, k;

    printf("밑을 입력하시오: ");
    scanf("%d", &x);
    printf("지수를 입력하시오: ");
    scanf("%d", &n);

    k = power(x, n);
    printf("%d의 %d제곱은 %d", x, n, k);
}

power(int x, int n) {
    if (n == 0) return 1;
    else return (x * power(x, n - 1));
}
```

① 5번
② 6번
③ 7번
④ 8번

정 답 : ②

정답해설

메인함수에서 power(3, 5)를 실행 -> 3*power(3, 4) 실행 -> 3*3*power(3, 3) 실행 -> 3*3*3*power(3, 2) 실행 -> 3*3*3*3*power(3, 1) 실행 -> 3*3*3*3*3*power(3, 0) 실행 -> 3*3*3*3*3*1 이 되므로 총 ② 6번 실행된다.

출제근거 도서명	재귀함수를 이해하고 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	118

(7) 다음은 원소의 논리적, 물리적 순서를 확인하기 위한 프로그램과 그것의 출력 값을 나타낸 것이다. 이때 num[1][1][2]의 원소와 주소를 구하시오.

```c
#include <stdio.h>

void main() {
    int *ptr, i;
    int num[3][2][4] = { {{30, 76, 99, 25}, {45, 17, 55, 92}},
                         { {4, 28, 7, 15}, {19, 83, 21, 11} },
                         { {22, 89, 47, 2}, {3, 9, 68, 44,} } };

    ptr = &num[0][0][0];
    printf("address: %u", ptr);
    return 0;
}
```

```
address: 10156824
```

① 7, 10156864
② 7, 10156868
③ 21, 10156880
④ 21, 10156884

정 답 : ③

정답해설

우선 초기화된 행렬인 num을 통해 num[1][1][2]의 원소가 21임을 알 수 있다.
그리고 프로그램의 출력 값을 통해 num[0][0][0]의 주소가 10156824임을 알 수 있으므로,
$\alpha + \{(i \times n_j \times n_k) + (j \times n_k) + k\} \times l$ 을 사용하여 주소를 구하면
$10156824 + \{(1 \times 2 \times 4) + (1 \times 4) + 2\} \times 4 = 10156880$ 이 된다. 따라서 답은 ③ 21, 10156880

출제근거 도서명	순차 자료구조를 이해하고 2차원 배열을 이용해 선형 리스트를 구현할 수 있으며 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	134

(8) 다음 희소행렬을 열 우선의 효율적인 2차원 배열로 나타낸 후, 전치행렬로 변환했을 때, A[2][5]의 원소를 구하시오.

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	11	0	28
[1]	25	0	0	0	0
[2]	0	8	0	0	43
[3]	0	0	0	19	0
[4]	13	0	39	0	0

① 0
② 4
③ 11
④ 39

정 답 : ④

정답해설

문제의 희소행렬을 열 우선의 효율적인 2차원 배열로 나타낸 후 전치행렬로 나타내면 다음과 같다. 따라서 A[2][5]의 원소는 ④ 39 이다.

	[0]	[1]	[2]
[0]	5	5	8
[1]	0	1	25
[2]	0	4	13
[3]	1	2	8
[4]	2	0	11
[5]	2	4	39
[6]	3	3	19
[7]	4	0	28
[8]	4	2	43

(a) 열 우선 순서 방법으로 나타낸 행렬

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
[0]	5	0	0	1	2	2	3	4	4
[1]	5	1	4	2	0	4	3	0	2
[2]	8	25	13	8	11	39	19	28	43

(b) (a)의 전치행렬

출제근거 도서명	선형 리스트를 이해하고 2차원 배열의 논리적, 물리적 순서를 확인할 수 있으며 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	143

(9) 다음은 노드를 pre 뒤에 삽입하는 연산을 나타낸 프로그램이다. 각 빈칸에 들어갈 코드로 알맞은 것을 구하시오. (단, linkedList_h는 리스트의 시작을 나타내는 구조체, listNode는 단순 연결 리스트의 노드구조를 구조체로 정의한 것이다.)

```
void InsertMiddleNode(linkedList_h* L, listNode* pre, char* x) {
    listNode* newNode;
    newNode = (listNode*)malloc(sizeof(listNode));
    strcpy(newNode->data, x);
    if (          ) {
        newNode->link = NULL;
        L->head = newNode;
    }
    else if (          ) {
        L->head = newNode;
    }
    else {
        newNode->link = pre->link;
        pre->link = newNode;
    }
}
```

① N == NULL, pre == NULL
② pre == NULL, N == NULL
③ N != NULL, pre != NULL
④ pre != NULL, N != NULL

정 답 : ①

정답해설

L이 공백 리스트인 경우에 새 노드를 첫 번째이자 마지막 노드로 연결해야 하므로 첫 번째 빈칸에 들어갈 코드는 N == NULL이다. 또, 삽입 위치를 지정하는 포인터 pre가 NULL인 경우 새 노드를 첫 번째 노드로 삽입해야 하므로 두 번째 빈칸에 들어갈 코드는 pre == NULL이다.
따라서 정답은 ① N == NULL, pre == NULL

출제근거 도서명	단순 연결 리스트를 이해하고 그것의 삽입과 삭제 연산을 할 수 있으며 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	169

(10) 다음은 원형 연결 리스트의 pre 뒤에 있는 노드 old를 삭제하는 연산을 나타낸 프로그램이다. 네모 칸 안의 코드는 무엇을 실행하기 위한 코드인가? (단, linkedList_h는 리스트의 시작을 나타내는 구조체, listNode는 원형 연결 리스트의 노드구조를 구조체로 정의한 것이다.)

```
void deleteNode(linkedList_h* CL, listNode* old) {
    listNode* pre;
    if (CL->head == NULL) return;
    if (CL->head->link == CL->head) {
        free(CL->head);
        CL->head = NULL;
        return;
    }
    else if (old == NULL) return;
    else {
        pre = CL->head;
        while (pre->link != old) {
            pre = pre->link;
        }
        pre->link = old->link;
        if (old == CL->head)
            CL->head = old->link;
        free(old);
    }
}
```

① 새 노드를 리스트의 시작 노드로 연결
② 삭제할 노드가 없는 경우 삭제 연산 중단
③ 공백 리스트인 경우 삭제 연산 중단
④ 리스트 시작 포인터를 NULL로 설정

정 답 : ③

정답해설

조건문 if (CL -> head == NULL) 는 CL이 가리키는 head가 NULL 일 때 즉 원형 연결 리스트가 공백리스트 일 때를 뜻하고 그 조건을 만족시키면 return 한다. 즉 공백 리스트인 경우 삭제 연산을 중단하는 것을 실행하긴 위한 코드이다. 따라서 답은 ③

(①, ④번은 조건문이 아니다. ②번 삭제할 노드가 없어서 연산을 중단하는 것이 아니라 리스트가 공백이기 때문에 연산 중단을 하는 것이다.)

출제근거 도서명	원형 연결 리스트를 이해하고 그것의 삽입과 삭제 연산을 할 수 있으며 이를 활용할 수 있다. c로 배우는 쉬운 자료구조, 이지영, 한빛아카데미, 2016	페이지	191