PART 10 포인터 연습문제 풀이

1. 다음 중 포인터가 나타내는 메모리 구획의 속성으로 가장 적합한 것은? ②

   ① 메모리 구획의 내용

   ② 메모리 구획의 주소

   ③ 메모리 구획의 크기

   ④ 메모리 구획의 종류

2. 다음 선언에 관해 바르게 설명한 것은? ②

```
int one = 1, *to_one = &one;
```

   ① 변수 one과 to_one을 모두 1로 초기화한 것이다.

   ② 변수 one은 1로 초기화하고 to_one은 one의 주소로 초기화한 것이다.

   ③ 변수 one은 1로 초기화하고 to_one의 주소를 one의 주소로 초기화한 것이다.

   ④ 변수 one은 1로 초기화한 것이 맞지만 to_one의 초기화는 잘못 초기화한 것이다.

3. 다음 중 포인터 연산에 관한 법칙으로 바른 것은? ③

   ① &(*a) == a

   ② *a == &a

   ③ a == *(&a)

   ④ &(*a) == *(&a)

4. 다음 함수 swap은 인수로 전달된 두 변수의 값을 서로 바꾸기 위한 함수다. swap에 대한 설명으로 바른 것은? ②

```
void swap(int a, int b) {
    a = a + b;
    b = a - b;
    a = a - b;
}
```

   ① swap은 매개변수 a, b의 값을 서로 바꾸지 못한다.

   ② swap은 매개변수 a, b의 값을 서로 바꾸지만 실인수로 전달된 값을 바꾸지 못한다.

   ③ swap을 이용하여 두 변수의 값을 바꾸려면 swap(&a, &b)와 같이 호출해야 한다.

   ④ swap은 매개변수 a, b의 값뿐 아니라 실인수로 전달된 값을 서로 바꾸지 못한다.

5. 다음과 같이 선언된 변수 a, b에 대하여 올바른 대입문은? ③

```
int a[5], *b;
```

① a = b;

② a = &b;

③ b = a;

④ b = *a;

6. 다음과 같이 선언된 변수 x의 타입에 대해 바른 설명은? ②

```
int *(x[10]);
```

① 크기 10인 int 형 배열을 가리키는 포인터

② 크기 10인 int* 형 배열

③ 크기 10인 int 형 배열을 받는 함수 포인터

④ int* 형 값을 반환하는 함수 포인터 10개로 구성된 배열

7. 다음과 같은 선언이 주어졌을 때 t와 같은 타입의 변수 s를 선언한 것은? ②

```
typedef int T[20];
T *t;
```

① int *s[20];

② int (*s)[20];

③ int *(s[20]);

④ int (*s[20])();

8. 다음 typedef 선언과 변수 선언이 주어졌다고 할 때, 각 변수를 일반 선언문으로 변경해 보아라.

```
typedef int A[10];
typedef double *P;
```

① A x[30];

int x[30][10];

② A *y;

int (*y)[10];

③ P z[20];

double *(z[20]);

9. 다음은 텍스트 파일을 한 줄씩 읽어 들인 다음 행 번호와 함께 출력해 주는 프로그램이다. 소문자를 대문자로 변경하여 출력하도록 프로그램을 변경하고자 한다. 한 라인을 받아서 그 라인 내의 문자들 중에서 소문자를 대문자로 변경하는 함수 to_uppers를 작성하여 원하는 작업을 하도록 프로그램을 변경해 보아라.

```c
//*********************************************************************
//  lines.c
//  텍스트 파일을 열고 행 번호와 함께 출력해 주는 프로그램
//*********************************************************************

#include <stdio.h>
#include <stdlib.h>

#define MAX_LINE        256

void print_with_lines(FILE *);

int main(int argc, char *argv[])
{
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s filename\n", argv[0]);
        exit(-1);
    }
    fp = fopen(argv[1], "r");
    if (!fp) {
        fprintf(stderr, "%s: cannot open file %s\n", argv[0], argv[1]);
        exit(-1);
    }
    print_with_lines(fp);

    return 0;
}

void print_with_lines(FILE *fp)
{
    char line[MAX_LINE];
    int line_num = 1;

    while (fgets(line, MAX_LINE, fp) != NULL) {
        printf("%6d: %s", line_num, line);
        line_num ++;
    }
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```

```c
#define MAX_LINE        256

void print_with_lines(FILE *);
void to_uppers(char *line);

int main(int argc, char *argv[])
{
    FILE *fp;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s filename\n", argv[0]);
        exit(-1);
    }
    fp = fopen(argv[1], "r");
    if (!fp) {
        fprintf(stderr, "%s: cannot open file %s\n", argv[0], argv[1]);
        exit(-1);
    }
    print_with_lines(fp);

    return 0;
}

void print_with_lines(FILE *fp)
{
    char line[MAX_LINE];
    int line_num = 1;

    while (fgets(line, MAX_LINE, fp) != NULL) {
        to_uppers(line);
        printf("%6d: %s", line_num, line);
        line_num ++;
    }
}

void to_uppers(char *p)
{
    while (*p) {
        *p = toupper(*p);
        p++;
```

```
    }
}
```

10. 프로그래밍 실습문제 2번으로 주어졌던 증분 k 알고리즘으로 암호화된 파일을 표준 입력으로 받아서 해독(decode)하는 프로그램을 작성하라. 역시 사용자로부터 패스워드 k를 입력받은 후 해독한다.

증분 k 알고리즘으로 암호화된 내용을 해독하려면 26 – k를 키로 하여 암호화 과정을 한 번 더 거치면 된다. 이 방법대로 작성한 프로그램은 다음과 같다.

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

void encode(char *p, int k);
void decode(char *p, int k);

int main()
{
    int k = 0;
    const int SZ = 256;
    char line[SZ];

    scanf("%d", &k);
    assert(k >= 0 && k <= 25);
    fgets(line, SZ, stdin);
    while (fgets(line, SZ, stdin) != NULL) {
        decode(line, k);
        printf("%s", line);
    }

    return 0;
}

void enc(char *p, int k)
{
    char c = *p;
    const int width = 'z' - 'a' + 1;
    int start = (islower(c))? 'a': 'A';
```

```
    if (isalpha(c)) {
        c -= start;
        c += k;
        c %= width;
        c += start;
    }
    *p = c;
}

void encode(char *p, int k)
{
    while (*p)
        enc(p++, k);
}

void decode(char *p, int k)
{
    int width = 'z' - 'a' + 1;
    k = width - k;
    encode(p, k);
}
```