

Report 6

Ch. 10 - 사용자 정의 함수와 라이브러리

학번	2018212236
학부	전자정보통신공학
이름	김동주

제출일자	2018-10-21
담당교수	반상우

작업환경	Visual Studio Code
컴파일러	MinGW GCC

필수 문제

1. Lotto

내용	아래 기능 포함한 Lotto 프로그램 구현 - 반복 기능 - Lotto 번호 중복 방지 기능
코드	<pre>#include <stdio.h> #include <stdlib.h> #include <stdbool.h> #include <time.h> #define LOTTERY_COST 1000 void __init__(); void autoGenerate(int Lottery[]); bool hasElement(int array[], int length, int item); void buyLottery(int Lottery[]); void getGradeOf(int Lottery[]); int USER[6]; int HOST[6]; int main(void) { char sel; __init__(); while(true) { buyLottery(USER); autoGenerate(HOST); getGradeOf(USER); printf("계속하시겠습니까? (y/n) : "); scanf(" %c", &sel);</pre>

```

        if (sel == 'n')
            break;
    }
}

void __init__() {
    long second = (long) time(NULL);
    srand(second);
}

void autoGenerate(int Lottery[]) {
    int num[45];
    int num_len = 45;

    for (int i = 0; i < 45; i++)
        num[i] = i + 1;

    int r;
    for (int i = 0; i < 6; i++) {
        r = rand() % num_len;

        Lottery[i] = num[r];

        for (int i = r; i < num_len - 1; i++)
            num[i] = num[i + 1];

        num_len--;
    }
}

bool hasElement(int array[], int length, int item)
{
    for (int i = 0; i < length; i++) {
        if (array[i] == item)
            return true;
    }
    return false;
}

```

```
// 게임 흐름 제어
```

```
void buyLottery(int Lottery[]) {  
    puts("복권을 구입합니다. (0을 입력하면 자동생성)");  
    for (int i = 0; i < 6; i++) {  
        printf(" [%d] : ", i + 1);  
        scanf("%d", &Lottery[i]);  
  
        if (i == 0 && Lottery[i] == 0) {  
            autoGenerate(USER);  
            break;  
        }  
  
        if (Lottery[i] < 1 || Lottery[i] > 45) {  
            puts("잘못된 입력입니다.");  
            i--;  
            continue;  
        }  
  
        if (hasElement(Lottery, i, Lottery[i])) {  
            puts("중복된 입력입니다.");  
            i--;  
            continue;  
        }  
    }  
  
    printf("구입한 복권번호\t: ");  
    for (int i = 0; i < 6; i++) {  
        printf("[%2d] ", USER[i]);  
    }  
    printf("\n");  
}
```

```
void getGradeOf(int Lottery[]) {  
    int matches = 0;  
  
    printf("이번회 당첨번호\t: ");  
    for (int i = 0; i < 6; i++) {  
        printf("[%2d] ", HOST[i]);  
    }
```

```
for (int j = 0; j < 6; j++) {
    if (i == j) continue;

    if (Lottery[i] == HOST[j]) {
        matches++;
        break;
    }
}

puts("\n");

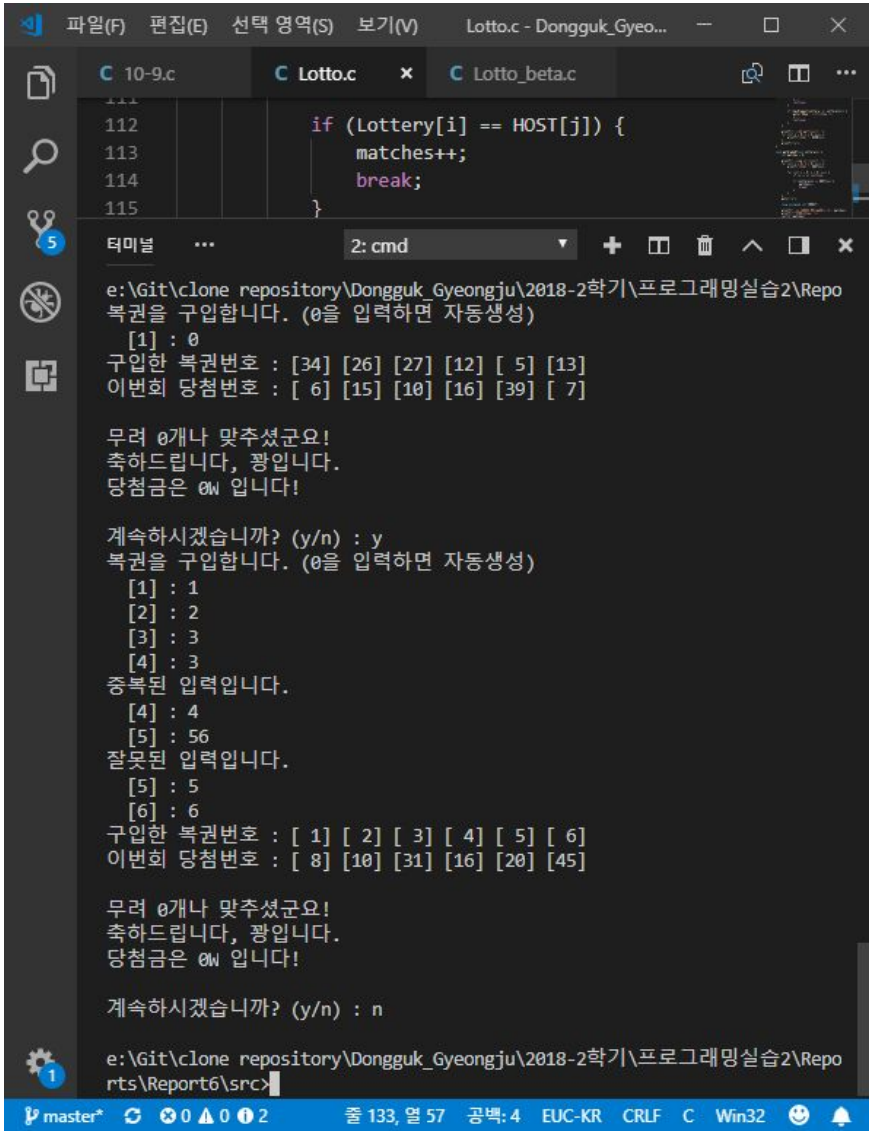
long unsigned int CREDIT;

printf("무려 %d개나 맞추셨군요!\n", matches);
printf("축하드립니다, ");
switch (matches)
{
    case 6: // 1등
        puts("1등입니다.");
        CREDIT = LOTTERY_COST * (long unsigned
int) 8145060;
        break;

    case 5: // 2등
        puts("2등입니다.");
        CREDIT = LOTTERY_COST * (long unsigned
int) 35724;
        break;

    case 4: // 3등
        puts("3등입니다.");
        CREDIT = LOTTERY_COST * 733;
        break;

    case 3: // 4등
        puts("4등입니다.");
        CREDIT = LOTTERY_COST * 45;
        break;
```

	<pre> default: puts("꽝입니다."); CREDIT = 0; break; } printf("당첨금은 %ld₩ 입니다!\n", CREDIT); puts(""); } </pre>
<p>실행결과</p>	 <pre> e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Repo 복권을 구입합니다. (0을 입력하면 자동생성) [1] : 0 구입한 복권번호 : [34] [26] [27] [12] [5] [13] 이번회 당첨번호 : [6] [15] [10] [16] [39] [7] 무려 0개나 맞추셨군요! 축하드립니다, 꽝입니다. 당첨금은 0₩ 입니다! 계속하시겠습니까? (y/n) : y 복권을 구입합니다. (0을 입력하면 자동생성) [1] : 1 [2] : 2 [3] : 3 [4] : 3 중복된 입력입니다. [4] : 4 [5] : 56 잘못된 입력입니다. [5] : 5 [6] : 6 구입한 복권번호 : [1] [2] [3] [4] [5] [6] 이번회 당첨번호 : [8] [10] [31] [16] [20] [45] 무려 0개나 맞추셨군요! 축하드립니다, 꽝입니다. 당첨금은 0₩ 입니다! 계속하시겠습니까? (y/n) : n e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Repo rts\Report6\src> </pre>
<p>고찰</p>	<p>위 문제를 풀이함에 있어서 가장 생각을 많이 했던 부분은 역시 중복방지이지 않을까 한다.</p> <p>중복을 방지하는 방법에는 여러가지가 있지만, 그 중 내가 사용하려고 하는 방법은 다음의 2가지이다.</p>

- 매번 입력 받을 때마다, 이전에 동일한 값을 입력 받은 적이 있는지 검사한다.
- 입력이 가능한 숫자를 하나씩 가지는 배열을 선언하고, 그 값들 중 랜덤으로 하나의 값을 선택해 가져오고, 배열에서 선택된 값을 삭제한다. (카드뽑기와 동일)

전자의 경우에는 웬만한 경우 빠르고, 구현하기가 쉽겠지만, 처리 속도가 안정적이지 않다. (운에 기대야 한다)

예로 다음과 같은 코드가 있을 때,

```
int r;
for (int i = 0; i < 6; i++) {
    r = rand() % num_len;

    if (hasElement(Lottery, i, r)) {
        puts("중복된 입력입니다.");
        i--;
        continue;
    }
}
```

r이 우연하게도 반복하여 같은 숫자가 나온다면 위의 코드는 무한 루프가 되어버린다.

(랜덤이기에 나올 가능성이 전혀 없는 것이 아니다. 희박하지만 이론상 가능하기는 하다)

반면, 후자인 카드뽑기의 방법에서는

```
void autoGenerate(int Lottery[]) {
    int num[45];
    int num_len = 45;

    for (int i = 0; i < 45; i++)
        num[i] = i + 1;

    int r;
    for (int i = 0; i < 6; i++) {
        r = rand() % num_len;

        Lottery[i] = num[r];
    }
}
```

```
for (int i = r; i < num_len - 1; i++)
    num[i] = num[i + 1];

num_len--;
}
```

한번 뽑아버린 카드 (선택된 숫자)는 (배열에서) 버려지기 때문에, 절대로 중복이 발생하지 않는다. 그렇기에 처리 속도가 매우 안정적이라는 장점이 있다.

다만, 요구되는 메모리가 많고, 경우에 따라서는 처리속도가 더 느릴 수도 있다.

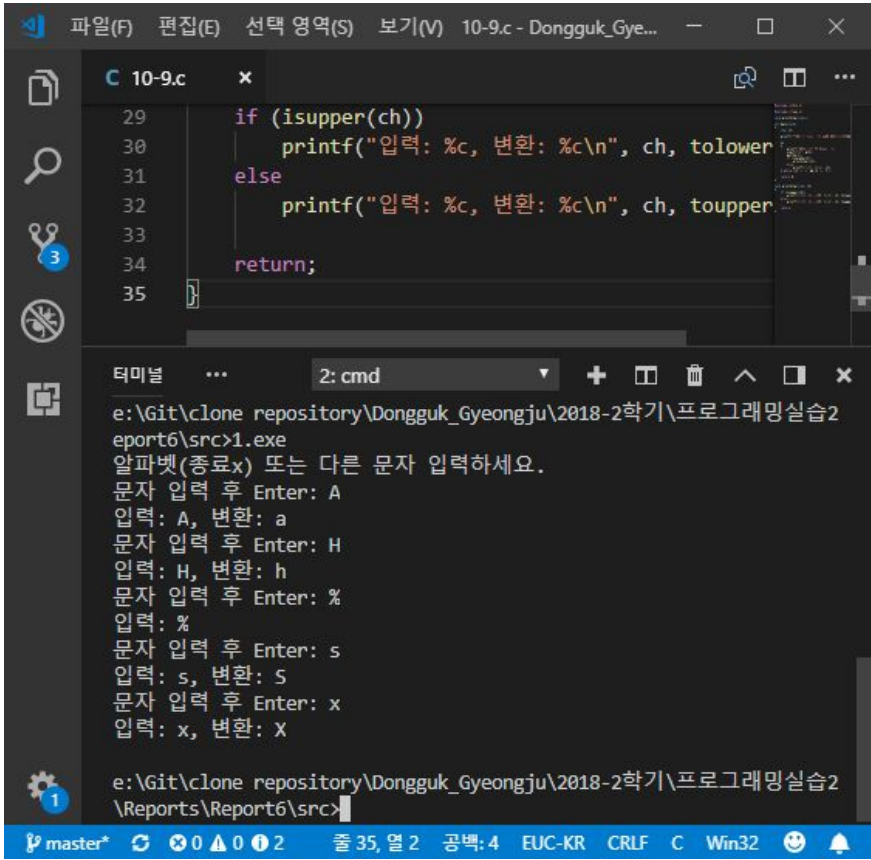
<느낀점>

결과적으로 같은 출력을 하는 코드라도, 그 과정에 따라서 (비동기적/동기적 작동이 요구됨에 따라, 혹은 고 스펙 장치/저 스펙 장치에서 실행됨에 따라서) 알맞게 사용해야 할 알고리즘이 다를 것이다.

그렇기에 고급 프로그래머가 되려면 알고리즘/자료구조에 대한 지식은 필수일 것이라고 생각하였다.

2. 실습 예제 10-9

내용	문자를 입력 받아 알파벳 문자이면 입력한 문자와 대소문자를 변환한 문자를 출력하는 프로그램이다.
코드	<pre>#include <stdio.h> #include <ctype.h> void print2char(char); int main(void) { char ch; printf("알파벳(종료x) 또는 다른 문자 입력하세요.\n"); do { printf("문자 입력 후 Enter: "); scanf("%c", &ch); getchar(); if (isalpha(ch)) print2char(ch); else printf("입력: %c\n", ch); } while (ch != 'x' && ch != 'X'); return 0; } void print2char(char ch) { if (isupper(ch)) printf("입력: %c, 변환: %c\n", ch, tolower(ch)); else printf("입력: %c, 변환: %c\n", ch, toupper(ch)); }</pre>

	<pre> return; } </pre>
실행결과	 <p>The screenshot shows a C program in a file named 10-9.c. The code uses the ctype.h library functions isupper, tolower, and toupper. The program prompts the user to enter a character and then displays its converted form. The terminal output shows the following sequence of inputs and outputs:</p> <pre> e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실습2\Report6\src>1.exe 알파벳(종료x) 또는 다른 문자 입력하세요. 문자 입력 후 Enter: A 입력: A, 변환: a 문자 입력 후 Enter: H 입력: H, 변환: h 문자 입력 후 Enter: % 입력: % 문자 입력 후 Enter: s 입력: s, 변환: S 문자 입력 후 Enter: x 입력: x, 변환: X </pre>
고찰	<p>ctype.h 라는 라이브러리를 사용하는 예제였다.</p> <p>위 예제에서 사용된 함수들인</p> <ul style="list-style-type: none"> • isalpha() • tolower() • toupper() <p>은 ctype.h안에서 원형이 선언되어 있을 것이다.</p> <p><Header파일에 관하여></p> <p>항상 c언어 프로그래밍을 하면서 가졌던 2가지 생각이 있었다.</p> <ol style="list-style-type: none"> 1. 헤더 파일은 어떤 원리로 작동하는가. 2. 대규모 프로젝트의 경우 소스코드 파일의 개수가 여러개가 사용되는 경우가 많은데, c언어는 한 프로젝트 안에 어떻게 다수의 소스코드를 연결할 수 있는가.

1.번에 관해서, 평소 의문만 하면서 직접 찾아보지 않았기에 잘 모르고 있었고, 막연하게 ‘헤더 파일은 기계어로 작성되어있지 않을까?’ 라고 생각하고 있었다.

하지만 최근 2학년인 친구의 ‘마이크로 프로세서 및 실습’의 과제를 함께 해보면서 직접 헤더파일을 작성할 일이 있었는데, 이 때 헤더파일 역시 C언어의 문법으로 작성되는 단순한 소스파일이었다는 점과, 우리가 자주 사용하는

- #include <파일이름>

의 방법은 내장 라이브러리를,

- #include “상대경로 혹은 절대경로/파일이름”

을 사용하면 유저 커스텀 헤더를 불러올 수 있다는 사실을 알게 되었다. (그리고 사용해보기도 하였다.)

이 방법을 활용하면 충분히 대규모 프로젝트에서 다수의 소스코드 파일로 나누어 작업할 수 있을 것이고, 유지보수 역시 쉬워질 것이라고 생각하였다.

P.S. 해당 마이크로 프로세서 실습 과제의 내 풀이 :

<https://gist.github.com/Hepheir/39ba3c61a76f3f852016ddce539e47c9>

[SW 9, 13 (5, 8번 스위치) 입력에 따라 LED 2~7 혹은 그 역순으로 ON, OFF하기 (파도타기)]

언젠가 왜 main 함수는 int로 작성되고, 여러 기능에 사용되는 정수형 변수로 int형을 사용하는가에 대해 “int가 가장 빠르기 때문이다” 라고 답변 받았던 적이 있었다.

그 때에는 아직 자세한 이유에 대해 배우기엔 나의 실력이 모자라다고 느꼈고 ‘그냥 그렇구나’하고 넘겼는데, 위의 마이크로 프로세서 실습을 하면서 함수도, 변수도 unsigned char 자료형을 사용하게 되었고, 예전에 했던 위의 질문이 떠오르면서 다음과 같은 생각을 하게 되었다.

- Windows의 경우 보편적으로 사용되는 것이 32bit/64bit 운영체제고, 32bit라 함은 4byte인데, 이는 32bit 운영체제에서 사용하는 int 자료형의 size 와 같다.
- 마이크로프로세서에서 사용하는 기판은 8bit 시스템을 사용하는 것으로 알고있는데, 여기서 8bit는 1byte이고 이는 char/unsigned char과 같은 크기이다.

=> 각 운영체제에서 사용하는 bit값과 같은 크기의 자료형이 가장 빠르지 않을까.

그럴 수 있는 것이, 8bit단위로 데이터를 처리하는 시스템에서 1byte가 아닌 값을 처리하려면, 일련의 처리과정을 거쳐야 할 테니 말이다.

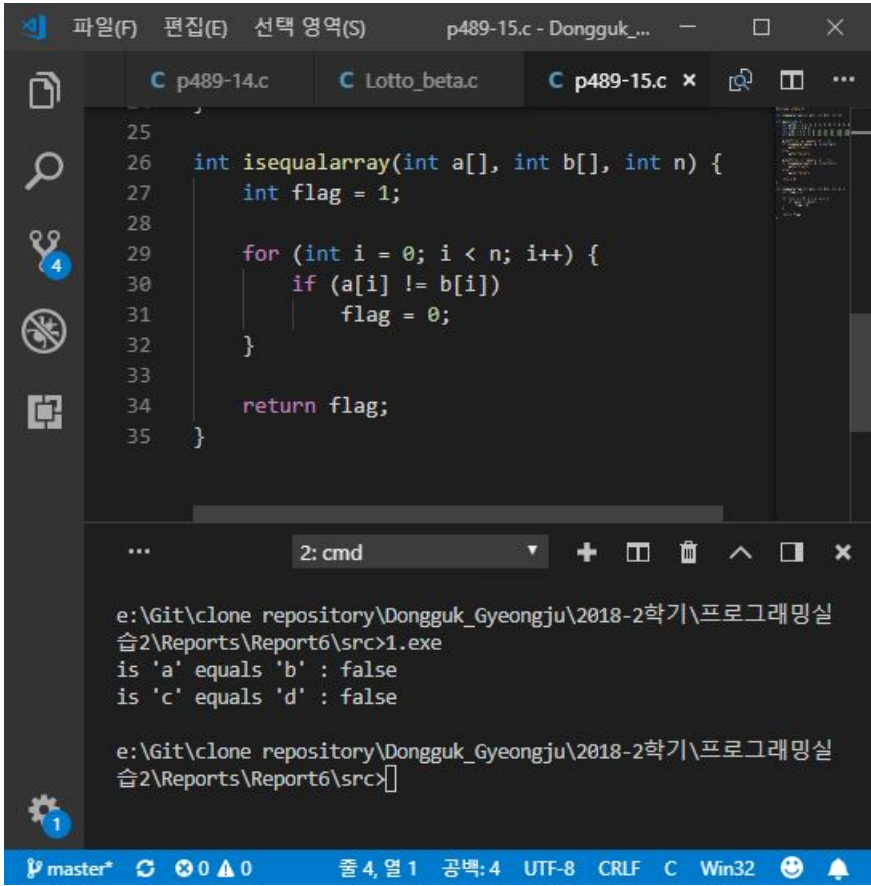
3. p.489-14

내용	<p>다음과 같이 일차원 배열을 복사하는 함수를 작성하여 결과를 알아보는 프로그램을 작성하시오.</p> <ul style="list-style-type: none">• void copyarray(int from[], int to[], int n /* 배열 원소 수 */)• 배열 from의 첫 번째 원소부터 (n-1)번째 원소까지 같은 순서대로 배열 to로 값을 복사하는 함수
코드	<pre>#include <stdio.h> void copyarray(int*, int*, int); int main(void) { int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 }; int b[10]; copyarray(a, b, sizeof(a)); for (int i = 0; i < 10; i++) printf("%d ", b[i]); puts(""); return 0; } void copyarray(int from[], int to[], int length) { for (int i = 0; i < length; i++) to[i] = from[i]; }</pre>

<p>실행결과</p>	 <pre> 1 #include <stdio.h> 2 3 void copyarray(int*, int*, int); 4 5 int main(void) { 6 int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 }; 7 8 copyarray(a, b, sizeof(a)); 9 10 for (int i = 0; i < 10; i++) 11 printf("%d ", b[i]); 12 13 puts(""); 14 return 0; </pre> <pre> e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실 e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실 e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실 습2\Reports\Report6\src>1.exe 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실 습2\Reports\Report6\src> </pre>
<p>고찰</p>	<p>Report 4, 5때 작성했던 코드와 같은 논리로,</p> <pre> void copyarray(int from[], int to[], int n); ... void copyarray(int*, int*, int); </pre> <p>위 부분 말고는 의도하지 않은 실행을 만들어 낼 것이라 의심되는 부분이 없는데, 출력결과를 보면 main함수에서 이루어져야 할 출력이 2번씩 반복되어 나오고 있다.</p> <p>코드 상에서는 main함수가 2번 호출되지 않은 것 같고, 아마 컴파일러 문제가 아닐까 생각이 되기도 한다.</p> <p>중간고사가 끝난 후 시간이 되면 위와 같은 현상이 일어나는 이유를 찾아봐야겠다.</p>

4. p.489-15

내용	<p>다음과 같이 일차원 배열의 동등함을 검사하는 함수를 작성하여 결과를 알아보는 프로그램을 작성하시오.</p> <ul style="list-style-type: none"> • int isequalarray(int a[], int b[], int n /* 배열 원소 수 */) • 배열 a와 b의 배열크기가 모두 n이며 순차적으로 원소 값이 모두 같으면 1을 반환, 아니면 0을 반환 하는 함수
코드	<pre>#include <stdio.h> int isequalarray(int a[], int b[], int n); int main(void) { int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 }; int b[10] = { 0 }; int c[8] = { 2, 4, 8, 16, 32, 64, 128, 256 }; int d[8] = { 2, 4, 8, 16, 32, 64, 128, 256 }; printf("is 'a' equals 'b' : "); if (isequalarray(a, b, sizeof(a))) puts("true"); else puts("false"); printf("is 'c' equals 'd' : "); if (isequalarray(c, d, sizeof(c))) puts("true"); else puts("false"); return 0; } int isequalarray(int a[], int b[], int n) { int flag = 1; for (int i = 0; i < n; i++) { if (a[i] != b[i]) flag = 0; } }</pre>

	<pre> return flag; } </pre>
실행결과	 <p>The screenshot shows a C++ IDE with the following code in <code>p489-15.c</code>:</p> <pre> 25 26 int isequalarray(int a[], int b[], int n) { 27 int flag = 1; 28 29 for (int i = 0; i < n; i++) { 30 if (a[i] != b[i]) 31 flag = 0; 32 } 33 34 return flag; 35 } </pre> <p>The terminal output shows the execution of the program:</p> <pre> e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실 습2\Reports\Report6\src>1.exe is 'a' equals 'b' : false is 'c' equals 'd' : false e:\Git\clone repository\Dongguk_Gyeongju\2018-2학기\프로그래밍실 습2\Reports\Report6\src> </pre>
고찰	<p><code>isequalarray()</code> 함수에서 지난 Report들과는 다른 방법을 사용해보았다.</p> <p>수업 중 교수님께서 언급하셨던 flag 기법을 사용하는 것이다.</p> <p>반환 값을 <code>flag = 1</code>, 특별한 일이 없는 한 참을 반환하는 변수를 두고, 두 배열을 검사한다.</p> <p>검사 도중 두 배열이 일치하지 않는 부분이 발견되면 <code>flag</code>의 값은 즉시 0 : 거짓이 되고, 마지막에 <code>return</code> 되는 값인 <code>flag</code>는 0이게 된다.</p> <p>이러한 flag 기법은 복잡하게 중첩되어있는 반복문 속에서 특정 조건을 만족할 시, 반복문 전체, 혹은 원하는 일부만 탈출/스킵 할 수 있게 해야할 때 매우 유용할 것이다.</p>

선택 문제

1. Lotto+

내용	<p>(필수 기능) 아래 기능 포함한 Lotto 프로그램 구현</p> <ul style="list-style-type: none">- 반복 기능- Lotto 번호 중복 방지 기능 <p>위의 필수 기능에 아래 기능을 추가로 포함한 Lotto 프로그램 구현</p> <ul style="list-style-type: none">- 사용자 번호 자동생성/직접입력 선택 기능- 그 외 추가 기능(실제 Lotto 와 동일하게 Bonus 번호를 활용한 2등 결정 등)
코드	<pre>#include <stdio.h> #include <stdlib.h> #include <stdbool.h> #include <time.h> #define LOTTERY_COST 1000 #define BONUS_MODE true void __init__(); void autoGenerate(int Lottery[], bool isBonus); bool hasElement(int array[], int length, int item); void buyLottery(int Lottery[]); void getGradeOf(int Lottery[]); int USER[6]; int HOST[6 + 1]; int main(void) { char sel; __init__(); while(true) { buyLottery(USER);</pre>


```

        autoGenerate(HOST, true);
        getGradeOf(USER);

        printf("계속하시겠습니까? (y/n) : ");
        scanf(" %c", &sel);

        if (sel == 'n')
            break;
    }
}

void __init__() {
    long second = (long) time(NULL);
    srand(second);
}

void autoGenerate(int Lottery[], bool isBonus) {
    int num[45];
    int num_len = 45;

    for (int i = 0; i < 45; i++)
        num[i] = i + 1;

    int r;
    for (int i = 0; i < 6; i++) {
        r = rand() % num_len;

        Lottery[i] = num[r];

        for (int i = r; i < num_len - 1; i++)
            num[i] = num[i + 1];

        num_len--;
    }

    if (isBonus) {
        r = rand() % num_len;
        Lottery[6] = num[r];
    }
}

```

```

}

bool hasElement(int array[], int length, int item)
{
    for (int i = 0; i < length; i++) {
        if (array[i] == item)
            return true;
    }
    return false;
}

// 게임 흐름 제어

void buyLottery(int Lottery[]) {
    puts("복권을 구입합니다. (0을 입력하면 자동생성)");
    for (int i = 0; i < 6; i++) {
        printf(" [%d] : ", i + 1);
        scanf("%d", &Lottery[i]);

        if (i == 0 && Lottery[i] == 0) {
            autoGenerate(USER, false);
            break;
        }

        if (Lottery[i] < 1 || Lottery[i] > 45) {
            puts("잘못된 입력입니다.");
            i--;
            continue;
        }

        if (hasElement(Lottery, i, Lottery[i])) {
            puts("중복된 입력입니다.");
            i--;
            continue;
        }
    }

    printf("구입한 복권번호\t: ");
    for (int i = 0; i < 6; i++) {

```

```

        printf("[%2d] ", USER[i]);
    }
    printf("\n");
}

void getGradeOf(int Lottery[]) {
    int matches = 0;
    bool match_bonus = false;

    printf("이번회 당첨번호\t: ");
    for (int i = 0; i < 6; i++) {
        printf("[%2d] ", HOST[i]);

        for (int j = 0; j < 6; j++) {
            if (i == j) continue;

            if (Lottery[i] == HOST[j]) {
                matches++;
                break;
            }
        }
    }

    if (BONUS_MODE) {
        printf("+ [%2d]", HOST[6]);

        for (int i = 0; i < 6; i++)
            if (HOST[6] == Lottery[i]) {
                match_bonus = true;
                break;
            }
    }

    puts("\n");

    long unsigned int CREDIT;

    printf("무려 %d개나 맞추셨군요!\n", matches);
    printf("축하드립니다, ");
    switch (matches)
    {

```

```
        case 6: // 1등
            puts("1등입니다.");
            CREDIT = LOTTERY_COST * (long unsigned
int) 8145060;
            break;

        case 5: // 2등
            if (match_bonus) {
                puts("2등입니다.");
                CREDIT = LOTTERY_COST * (long
unsigned int) 1357510;
            } else {
                puts("3등입니다.");
                CREDIT = LOTTERY_COST * (long
unsigned int) 35724;
            }
            break;

        case 4: // 3등
            puts("4등입니다.");
            CREDIT = LOTTERY_COST * 733;
            break;

        case 3: // 4등
            puts("5등입니다.");
            CREDIT = LOTTERY_COST * 45;
            break;

        default:
            puts("꽝입니다.");
            CREDIT = 0;
            break;
    }
    printf("당첨금은 %ld₩ 입니다!\n", CREDIT);
    puts("");
}
```

실행결과

The screenshot shows a C program named 'Lotto_beta.c' being executed. The code in the editor includes a loop to compare user input with lottery numbers and a bonus check. The terminal output shows the program's execution flow, including the purchase of a ticket, the generated numbers, the winning numbers, and the final result of a 2nd prize win.

```

119         if (Lottery[i] == HOST[j]) {
120             matches++;
121             break;
122         }
123     }
124 }
125 if (BONUS_MODE) {
126     printf("+ [%2d]", HOST[6]);
127 }
128 for (int i = 0; i < 6; i++)
129     if (HOST[6] == Lottery[i]) {
130         match_bonus = true;
131         break;

```

... 2: 1

복권을 구입합니다. (0을 입력하면 자동생성)
 [1] : 0
 구입한 복권번호 : [14] [27] [11] [43] [25] [22]
 이번회 당첨번호 : [2] [45] [41] [22] [19] [1] + [3]

 무려 1개나 맞추셨군요!
 축하드립니다, 팡입니다.
 당첨금은 0₩ 입니다!

 계속하시겠습니까? (y/n) : []

master* 줄 174, 열 2(3738 선택됨) 공백: 4 EUC-KR CRLF C Win32

고찰

<사용자 번호 자동생성/직접입력 선택 기능>

필수 문제 풀이에서 당첨번호의 뽑기 기능을 autoGenerate() 라는 함수로 분리하여 사용했기 때문에, 사용자가 자동생성 기능을 사용하게 하는 것이 매우 간단했다.

```

void buyLottery(int Lottery[]) {
    puts("복권을 구입합니다. (0을 입력하면
    자동생성)");
    for (int i = 0; i < 6; i++) {
        printf(" [%d] : ", i + 1);
        scanf("%d", &Lottery[i]);

        if (i == 0 && Lottery[i] == 0) {
            autoGenerate(USER, false);
            break;
        }
    }
    ...
}

```

<그 외 추가 기능(실제 Lotto 와 동일하게 Bonus 번호를 활용한 2등

결정 등)>

Bonus 번호를 이용하기위해 HOST배열의 크기를 1늘여 7개의 번호를 가지도록하고, getGradeOf() 함수에서

```
if (BONUS_MODE) {
    printf("[%2d]", HOST[6]);

    for (int i = 0; i < 6; i++)
        if (HOST[6] == Lottery[i]) {
            match_bonus = true;
            break;
        }
}
```

위의 조건문을 통해 보너스공을 맞추었는지 여부를 판단한다.

```
case 5: // 2등
    if (match_bonus) {
        puts("2등입니다.");
        CREDIT = LOTTERY_COST * (long unsigned
int) 1357510;
    } else {
        puts("3등입니다.");
        CREDIT = LOTTERY_COST * (long unsigned
int) 35724;
    }
    break;
```

후에, 기존 2등에서 보너스 공을 맞추지 못했으면 3등으로 차등을 두었다.

P.S. 필수 1번 문제 풀이를 할 때, 당첨금인 CREDIT을 구하는 방법은, 복권 1개의 가격이 LOTTERY_COST이고, 가격에 당첨 확률 (위키에서 참고함)을 곱하는 것으로 하였다.

그러다보니 1등의 당첨 액수가 터무니없이 커졌고, int형의 표현 범위를 넘어가기까지 하였다. 따라서 CREDIT을 구하는데에는 long unsigned int 자료형을 사용하였다.

<느낀점>

필수 문제에서 작성하였던 코드에서, 번호 자동생성 기능을

	<p>autoGenerate() 함수로 분리해두었기에, 동일한 기능을 추가해야 할 때, 매우 편리했다.</p> <p>코드를 작성할 때, 유지 보수성을 염두해두고 위와 같이 코딩을 했을 때, 실제로 편리하다는 것을 이 문제 풀이를 통해 느끼게 되었다.</p>
--	---