

과제 1

[과제 설명]

어떤 수 x 가 1보다 큰 제곱수로 나누어 떨어지지 않을 때, 제곱ㄴㄴ수라고 한다. 제곱수는 4, 9, 16, 25와 같은 것이고, 제곱ㄴㄴ수는 1, 2, 3, 5, 6, 7, 10, 11, 13, ...과 같은 수이다. min과 max가 주어지면, min과 max를 포함한 사이에 제곱ㄴㄴ수가 몇 개 있는지 출력한다. 입력은 정수 1 ~ 10000사이로 한다.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main() {

    // 변수 선언하기.
    int min = 0, max = 0, squareCnt = 0;
    // 제곱 ㄴㄴ수를 담는 배열.
    int squareNumAry[10000] = { 0, };
    int i, j, isSquare = 0;

    // 형식 출력하기.
    printf("%min : ");
    scanf("%d", &min);
    printf("%max : ");
    scanf("%d", &max);

    // 제곱 ㄴㄴ수 구하기
    /*
    * 추정하는 공식 : (어떤 수 x) % (제곱 수) != 0 -> 제곱 ㄴㄴ수.
    * 어떤 수 x : min이상 ~ max 이하의 모든 수.
    * 제곱 수 : 4, 9, 16 ~~~.
    */

    for (i = min; i <= max; i++) {
        // 임의의 제곱 수 선언
        int tempSquareNum = 0;
        for (j = 2; j < 10000; j++) {
            tempSquareNum = j * j;
            // 제곱 ㄴㄴ수가 아니라면
            if (i % tempSquareNum == 0) {
                isSquare = 0;
                break;
            } else {
                isSquare = 1;
            }
        }
        // 제곱 수 증가 공식
    }
    if (isSquare == 1) {
        squareNumAry[squareCnt] = i;
        // 제곱 ㄴㄴ수가 배열에 포함되면 해당 배열의 index 값 증가시키기.
        squareCnt++;
    }
}
```

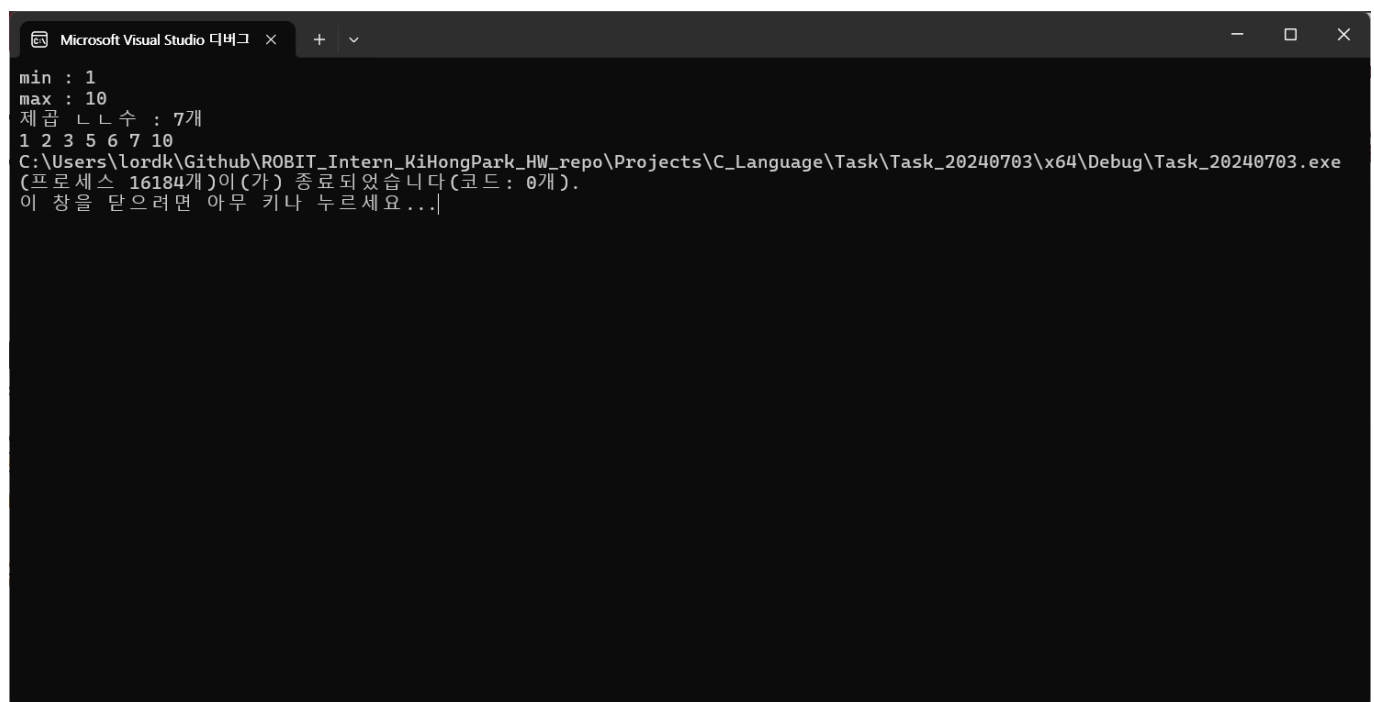
```
        isSquare = 0;
    }

    printf("제공 ㄴㄴ수 : %d개\n", squareCnt);

    // 제공 ㄴㄴ수가 담긴 배열을 출력하기
    for (i = 0; i < 10000; i++) {
        if (squareNumAry[i] == 0) {
            break;
        }else{
            printf("%d ", squareNumAry[i]);
        }
    }

    return 0;
}
```

Test Case #1



```
Microsoft Visual Studio 디버그
min : 1
max : 10
제공 ㄴㄴ수 : 7개
1 2 3 5 6 7 10
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\Task\Task_20240703\x64\Debug\Task_20240703.exe
(프로세스 16184개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

Test Case #2

```
Microsoft Visual Studio 디버그 x + v
min : 123
max : 123
제공된 수 : 1개
123
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\Task\Task_20240703\x64\Debug\Task_20240703.exe
(프로세스 22800개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

Test Case #3

```
Microsoft Visual Studio 디버그 x + v
min : 20
max : 100
제공된 수 : 48개
21 22 23 26 29 30 31 33 34 35 37 38 39 41 42 43 46 47 51 53 55 57 58 59 61 62 65 66 67 69 70 71 73 74 77 78 79 82 83 85
86 87 89 91 93 94 95 97
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\Task\Task_20240703\x64\Debug\Task_20240703.exe
(프로세스 8944개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

과제 2

[과제 설명]

비어있는 공집합 S 가 주어졌을 때, 아래 연산을 수행하는 프로그램을 작성하시오.

- add x : S 에 x 를 추가한다. ($1 \leq x \leq 20$) S 에 x 가 이미 있는 경우에는 연산을 무시한다.
- remove x : S 에서 x 를 제거한다. ($1 \leq x \leq 20$) S 에 x 가 없는 경우에는 연산을 무시한다.
- check x : S 에 x 가 있으면 1을, 없으면 0을 출력한다.
- toggle x : S 에 x 가 있으면 x 를 제거하고, 없으면 x 를 추가한다. ($1 \leq x \leq 20$)
- all 0: S 를 $\{1, 2, \dots, 20\}$ 으로 바꾼다.
- empty 0: S 를 공집합으로 바꾼다.

```

#define _CRT_SECURE_NO_WARNINGS
#include <string.h>
#include <stdio.h>

int main() {

    // 변수 선언하기
    int inputNum = 0, numAry[20] = { 0, }; // numAry : 요소를 저장하는 집합(배열)
    char operator[10] = { 0, }; // add, remove 등의 단어 저장 배열
    int i, j, aryIndex = 0;
    int isOkayAdd = 1, isOkayRemove = 0, isOkayCheck = 0, isOkayToggle = 0; //
    boolean 연산자는 안 배운 내용이므로 int로 boolean 역할을 대체함.

    // 형식 출력하기
    printf("연산을 선택하세요. (1 <= x <= 20");
    printf("\nadd X\nremove X\ncheck X\ntoggle X\nall 0\nempty 0\n\n");

    while (1) {

        printf("input : ");
        scanf("%s %d", &operator, &inputNum);

        /* (예외처리)
        * 과제에는 언제까지 반복하는지, 언제 탈출하는지에 대한 설명이 안 되어 있음.
        * 무한 루프를 방지하기 위해 반복문 탈출에 대한 예외처리는 임의로 구현함.
        * 단, 과제에 제시된 출력 형식을 준수하기 위해 형식 출력에서는 stop 관련 문구를
        추가하지 않음.
        * 물론, Ctrl + C 단축키를 통해 무한 루프를 탈출할 수 있음.
        */

        if (!strcmp(operator, "stop") && inputNum == 0) {
            printf("Stopped!");
            break;
        }

        // 계산 연산자 비교하기
        if (!strcmp(operator, "add")) {
            for (i = 0; i < 20; i++) {
                if (numAry[i] == inputNum) {
                    isOkayAdd = 0;
                    break;
                }
            }
            if (isOkayAdd == 1) {
                numAry[aryIndex] = inputNum;
                printf("집합 : { ");
                for (j = 0; j <= aryIndex; j++) {
                    printf("%d, ", numAry[j]);
                }
                printf(" }\n\n");
                aryIndex++;
            }
            isOkayAdd = 1;

```

```

}else if (!strcmp(operator, "remove")) {
    int removeIdx = 0; // 삭제할 요소의 index 값

    // 삭제할 수 있는 숫자인지 검토하기. (해당 숫자가 집합의 요소로 포함되어 있
    는지 검토)
    for (i = 0; i < 20; i++) {
        // 숫자가 집합 내에 있는 요소일 때의 처리.
        if (numAry[i] == inputNum) {
            removeIdx = i;
            aryIndex--; // 요소를 추가할 때, 변경된 index 업데이트하기
            /* ex) remove가 두 번 되었으면,
            * 추가할 때의 index 위치도 추가할 때의 index에서 두 번 뒤에서 추
            가해야 함.

            */
            isOkayRemove = 1;
            break;
        }
    }
    if (isOkayRemove == 1) {
        // 요소 제거하기.
        numAry[removeIdx] = 0;
        // 제거된 요소 뒤에 있는 요소들도 앞으로 이동 시키기.
        for (j = removeIdx; j < 20; j++) {
            numAry[j] = numAry[j + 1];
        }
    }
    // 디버깅 (요소 출력하기 - 정리 후)
    printf("집합 : { ");
    for (j = 0; j < aryIndex; j++) {
        printf("%d, ", numAry[j]);
    }
    printf(" }\n\n");
    isOkayRemove = 0;

}else if (!strcmp(operator, "check")) {
    // 입력된 숫자가 집합에 포함되어 있는지 검토하기
    for (i = 0; i < 20; i++) {
        if (numAry[i] == inputNum) {
            isOkayCheck = 1;
            break;
        }else{
            isOkayCheck = 0;
        }
    }
    // 만약, 집합 내에 해당 숫자가 없으면 0을 출력하기.
    // 배운 개념 중 하나인 *3항 연산자* 사용해 보기.
    printf("%d ", isOkayCheck == 1 ? 1 : 0);

    printf("집합 : { ");
    for (j = 0; j < aryIndex; j++) {
        printf("%d, ", numAry[j]);
    }
    printf(" }\n\n");

```

```

}else if (!strcmp(operator, "toggle")) {
    int removeIdx = 0;
    // 입력된 숫자가 집합에 포함되어 있는지 검토하기
    for (i = 0; i < 20; i++) {
        if (numAry[i] == inputNum) {
            removeIdx = i;
            isOkayToggle = 1;
            break;
        }else {
            isOkayToggle = 0;
        }
    }
    // 만약 집합에 해당 숫자가 있을 때,
    if (isOkayToggle == 1) {
        // 요소 제거하기.
        numAry[removeIdx] = 0;
        aryIndex--;
        // 제거된 요소 뒤에 있는 요소들도 앞으로 이동 시키기.
        for (j = removeIdx; j < 20; j++) {
            numAry[j] = numAry[j + 1];
        }
    }else{ // 만약 집합에 해당 숫자가 없을 때,
        numAry[aryIndex] = inputNum;
        aryIndex++;
    }

    printf("집합 : { ");
    for (j = 0; j < aryIndex; j++) {
        printf("%d, ", numAry[j]);
    }
    printf(" }\n\n");

}else if (!strcmp(operator, "all") && inputNum == 0) {
    // 과제에 제시된 입력 형식 "all 0"이 되어야만 제시된 집합으로 변경되도록 구
    현함.

    for (j = 0; j < 20; j++) {
        numAry[j] = j + 1;
    }

    printf("집합 : { ");
    for (j = 0; j < 20; j++) {
        printf("%d, ", numAry[j]);
    }
    printf(" }\n\n");
    aryIndex = 19;

}else if (!strcmp(operator, "empty") && inputNum == 0) {
    // 과제에 제시된 입력 형식 "empty 0"이 되어야만 제시된 집합으로 변경되도록
    구현함.

    /*
    * 요소는 0로 지정하여 공집합 처리를 함.
    * 0으로 지정한 이유 : 입력된 숫자의 범위는 1이상 ~ 20이하임.

```

```

* 집합 검사 식에서 0인 요소가 있으면 공집합임을 인식하게 하여,
* 공집합 출력을 하도록 구현함.
*/

for (j = 0; j < 20; j++) {
    numAry[j] = 0;
}

printf("집합 : { ");
for (j = 0; j < aryIndex; j++) {
    // 공집합이라면
    if (numAry[j] == 0) {
        break;
    }
}
printf(" }\n\n");
aryIndex = 0;
}

}

return 0;
}

```

Test Case #1

```

C:\Users\Wlordk\Github\WROE x + ~
연산을 선택하세요. (1 <= x <= 20)
add X
remove X
check X
toggle X
all 0
empty 0

input : add 5
집합 : { 5, }

input : add 3
집합 : { 5, 3, }

input : remove 5
집합 : { 3, }

input : check 3
1 집합 : { 3, }

input : toggle 3
집합 : { }

input : all 0
집합 : { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, }

input : empty 0
집합 : { }

input : |

```

Test Case #2

```
C:\Users\WlordkW\Github\WROE x + v
연산을 선택하세요. (1 <= x <= 20)
add X
remove X
check X
toggle X
all 0
empty 0

input : all 0
집합 : { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, }

input : add 2
input : remove 3
집합 : { 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, }

input : remove 19
집합 : { 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, }

input : toggle 11
집합 : { 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, }

input : toggle 11
집합 : { 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 11, }

input : toggle 19
집합 : { 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 11, 19, }

input : |
```

Test Case #3

```
C:\Users\WlordkW\Github\WROE x + v
연산을 선택하세요. (1 <= x <= 20)
add X
remove X
check X
toggle X
all 0
empty 0

input : add 3
집합 : { 3, }

input : add 9
집합 : { 3, 9, }

input : add 8
집합 : { 3, 9, 8, }

input : add 4
집합 : { 3, 9, 8, 4, }

input : empty 0
집합 : { }

input : add 3
집합 : { 3, }

input : check 6
0 집합 : { 3, }

input : |
```