1번 문제 제출완료

☑ 4점

Complete the given source code to build a program displaying a source code with the line number in front of each line.

Use "%2d" as the line number output FSF.

The name (including path) of the source code file is given as a string pointed by *filename*.

You can check the content of the target file using the "data/stdint-wrap.h".

입/출력 예시

␣ : 공백 ↵ : 줄바꿈 ⇥ : 탭

예시 1

출력

```
␣1␣:␣#ifndef␣_GCC_WRAP_STDINT_H↵
␣2␣:␣#if␣__STDC_HOSTED__↵
␣3␣:␣#␣if␣defined␣__cplusplus␣&&␣__cplusplus␣>=␣201103L↵
␣4␣:␣#␣␣undef␣__STDC_LIMIT_MACROS↵
␣5␣:␣#␣␣define␣__STDC_LIMIT_MACROS↵
␣6␣:␣#␣␣undef␣__STDC_CONSTANT_MACROS↵
␣7␣:␣#␣␣define␣__STDC_CONSTANT_MACROS↵
␣8␣:␣#␣endif↵
␣9␣:␣#␣include_next␣<stdint.h>↵
10␣:␣#else↵
11␣:␣#␣include␣"stdint-gcc.h"↵
12␣:␣#endif↵
13␣:␣#define␣_GCC_WRAP_STDINT_H↵
14␣:␣#endif↵
```

2번 문제 제출완료

☑ 4점

Refer fseek() and ftell() functions designed to control "File Positioning" in the C language. Complete the given source code to build a program finding out the size of a file using these standard library functions. The name of a target file is given as a command line argument.

(표준 라이브러리 함수인 fseek(), ftell()을 활용하여 파일의 크기를 구하는 프로그램을 작성하라. 대상 파일의 이름은 명령어 인자로 전달된다)

Refer to the following code to output the filesize.

printf("File %s size : %d\n", argv[1], filesize);

Your program should display a usage message when there is no argument as follows: (명령어 인자가 없으면 아래와 같이 사용법을 출력하여야 한다)

Usage : fsize <file>

USE "return 0" to terminate/end your main() for this problem. Using "exit(1)" or "return 1" will trigger unexpected results.

(이 문제에서 main() 함수를 종료할 때는 "return 0"를 사용하라. "exit(1)" 또는 "return 1" 등은 원치 않는 결과를 발생시킬 수 있다)

Consider the input in each test case as the corresponding command line arguments for the case. For example, if you just press "enter" without writing a file name, it is considered as no command line argument and your program should print out "Usage" message. If you enter "libgcc_s.so", it is considered a file name. The file names included in the test cases are "libgcc_s.so", "libgcc_eh.a" and "libgcc.a"

(각 테스트케이스에서 입력을 해당 테스트케이스의 명령어 인자로 간주하라. 예를 들어 프로그램 실행 후 아무런 입력 없이 단순히 "enter"를 누르면 명령어 인자가 없는 것으로 간주되며 프로그램은 "Usage" 메시지를 출력하여야 한다. 입력으로 "libgcc_s.so"를 입력하면 파일명으로 간주된다. 평가를 위한 테스트케이스에 포함된 파일의 이름은 "libgcc_s.so", "libgcc_eh.a", "libgcc.a"이다)

예시 1

입력

```
↵
```

출력

```
Usage␣:␣fsize␣<file>↵
```

예시 2

입력

```
libgcc_s.so
```

출력

```
File␣libgcc_s.so␣size␣:␣132↵
```

3번 문제 제출완료

☑ 4점

Complete the given source code to build a program calculating interest on term deposits explained in the lecture slide.

int inter_calc(int dep, double rate, int yrs);

/**

* @brief : Calculate the total interest on the deposit money for the entire deposit term. The total interest with the deposit money a, interest rate r, and deposit term n year(s) is $a \cdot (1 + r)^n - a, (n \geq 1)$. If the term is shorter than 1 year, the interest is 0.

* @return : The total interest rounded to the nearest integer

* @param : dep – the amount of money deposited

rate – interest rate per year (0.018 == 1.8%)

yrs – deposit term in years

*/

Your program should display a usage message when there is no argument as follows: (명령어 인자가 없으면 아래와 같이 사용법을 출력하여야 한다)

Usage: bankint <deposit_money> <interest_rate(%)> <term(year(s))>

Default interest = 1.8%, Default term = 1 year

USE "return 0" to terminate/end your main() for this problem. Using "exit(1)" or "return 1" will trigger unexpected results.

(이 문제에서 main() 함수를 종료할 때는 "return 0"를 사용하라. "exit(1)" 또는 "return 1" 등은 원치 않는 결과를 발생시킬 수 있다)

Consider the input in each test case as the corresponding command line arguments for the case.

(각 테스트케이스에서 입력을 해당 테스트케이스의 명령어 인자로 간주하라)

예시 1

입력

```
↵
```

출력

```
Usage:␣bankint␣<deposit_money>␣<interest_rate(%)>␣<term(year(s))>↵
Default␣interest␣=␣1.8%,␣Default␣term␣=␣1␣year↵
```

예시 2

입력

```
10000000↵
```

출력

```
Total␣Interest␣=␣KRW␣180000↵
```

예시 3

입력

```
10000000␣2.2␣2
```

출력

```
Total␣Interest␣=␣KRW␣444840↵
```

4번 문제 제출완료

☑ 4점

Complete the given source code to build a program generating non-negative random integers using rand() function. The number of random numbers and the seed value are given as command line arguments.

(rand() 함수를 이용하여 음이 아닌 정수 난수를 생성하는 프로그램을 작성하라. 생성해야할 난수의 수와 seed 값은 명령어 인자로 주어진다)

Enter a proper code to handle the command line arguments in the main() function. Refer to the "Usage" messages in the main() to understand how to handle command line arguments, N, and seed. As described in the "Usage", N is a mandatory/required argument and seed is an optional argument. The default value of the seed is 10.

(main() 함수에 명령어 인자를 처리하는 코드를 추가하라. main() 함수의 Usage 메시지를 보면 명령어 인자, N과 seed를 어떻게 처리할 지 이해할 수 있을 것이다. N은 필수 인자이며 seed는 선택적 인자이며 기본값은 10이다.)

You need to complete the generate_numbers() described below.

int * generate_numbers(int nnums);

/**

* @brief : Generate "nnums" non-negative random values using the rand() function and store them into a dynamically allocated integer array(memory) and return its address. To indicate the end of valid values or the end of the array, append a negative value after the last valid random value. For example, if nnums = 5 and the random numbers are 3, 5, 19, 0, 8, the dynamically allocated array should have values 3, 5, 19, 0, 8, -1.

(rand() 함수를 이용하여 nnums 개의 음이 아닌 난수 정수를 생성하고 이를 동적으로 할당한 정수 배열에 저장하고 그 주소를 반환하는 함수. 유효한 값의 끝, 즉 배열의 끝을 표시하기 위하여 마지막에 음의 값을 추가하라. 예를 들어 nnums=5이고 생성한 난수 값이 3,5,19,0,8 이라면 동적으로 할당된 배열은 3,5,19,0,8,-1 값을 가져야 한다.)

* @return : The pointer/address of the dynamically allocated memory storing random numbers.

* @param : nnums – the # of random numbers to generate

*/

Consider the input in each test case as the corresponding command line arguments for the case. For example, if you just press "enter" without writing any, it is considered as no command line argument and your program should print out "Usage" message. If you enter "10", it is considered a value corresponding to <N>

(각 테스트케이스에서 입력을 해당 테스트케이스의 명령어 인자로 간주하라. 예를 들어 프로그램 실행 후 아무런 입력 없이 단순히 "enter"를 누르면 명령어 인자가 없는 것으로 간주되며 프로그램은 "Usage" 메시지를 출력하여야 한다. 입력으로 "10"을 입력하면 인자 <N>에 해당하는 값으로 간주된다)

예시 1

입력

```
10␣↵
```

출력

```
1215069295␣1311962008␣1086128678␣385788725␣1753820418␣↵
394002377␣1255532675␣906573271␣54404747␣679162307␣↵
```

예시 2

입력

```
15␣1
```

출력

```
1804289383␣846930886␣1681692777␣1714636915␣1957747793␣↵
424238335␣719885386␣1649760492␣596516649␣1189641421␣↵
1025202362␣1350490027␣783368690␣1102520059␣2044897763␣↵
```

예시 3

입력

```
20␣2
```

출력

```
1505335290␣1738766719␣190686788␣260874575␣747983061␣↵
906156498␣1502820864␣142559277␣1261608745␣1380759627␣↵
2127304342␣635050179␣582691149␣149585093␣2039335037␣↵
820715049␣693014654␣2122498773␣1809302367␣591232730␣↵
```

예시 4

출력

```
Usage␣:␣generate_numbers␣<␣N␣>␣[<seed>]↵
↵
N␣:␣Mandatory,␣Number␣of␣random␣numbers␣to␣generate↵
seed␣:␣An␣optional␣seed␣value␣to␣seeds␣the␣random␣number␣generator,␣defalut␣=␣0
```

5번 문제 제출완료

4점

Complete the count_distinct_values() function counting the number of distinct values in an integer array.

You can add your own functions if necessary.

int count_distinct_values(int *values);

/**

* @brief : Count the number of distinct values in an integer array pointed by the parameter "values". All valid values in the array are non-negative and a negative value indicates the end of the array. Consider an integer array example, "test_values" defined in the main(), count_distinct_values(test_values) should return 6, as it has 6 distinct values, 1, 2, 3, 4, 5 and 7. "-1" at the end of the array is not a valid value and it is used just to indicate the end of the array.

(정수 배열이 인자로 주어질 때 해당 배열 내에 서로 다른 정수의 개수를 세는 함수이다. 배열의 유효 항목은 모두 음수가 아니다. 음수가 쓰일 경우 그것은 배열의 끝은 나타내기 위함이다. main()에 정의된 "test_values"라는 정수 배열 예를 살펴보라. 만약 count_distinct_values(test_values)가 호출되면 함수는 6을 반환하여야 한다. 이 배열 안에는 1, 2, 3, 4, 5 그리고 7이라는 6개의 서로 다른 수가 있기 때문이다. 배열의 마지막 값인 "-1"은 유효한 값이 아니며 단지 배열의 끝을 나타내기 사용되었다.)

* @return : The number of distinct values in the array.

* @param : values – the pointer to an integer array

*/

예시 1

입력

```
1↵
```

출력

```
#␣of␣distinct␣values␣=␣6↵
```

6번 문제 제출완료

☑ 4점

Complete **read_numbers_binary()** and **read_numbers_csv()** in the given source code reading non-negative integer values from the given file and returning the pointer of an integer array containing the read values. The integer array returned by the functions **needs to be allocated dynamically** inside the functions and "-1" should be appended after all valid values to indicate the end of the array.

(주어진 파일에서 음이 아닌 정수 값들을 읽어 들이는 read_numbers_binary(), read_numbers_csv() 함수를 완성하라. 이 함수는 읽어 들인 정수 값들을 모두 포함하고 있는 정수 배열의 포인터를 반환하여야 한다. 값을 저장하는 정수 배열들은 함수 내에서 동적으로 할당되어야 하며 모든 유효 값의 끝에 "-1" 값을 덧붙여 배열의 끝임을 표시하여야 한다.)

The file to be read by "read_numbers_binary()" is written through the function **"save_numbers_binary()"** included in the given source code. Similarly, the file to be read by "read_number_csv()" is written through the function **"save_numbers_csv()"**. To verify your function, "get_max()" function returning the maximum values in the array will be called. It is also included in the given source code.

(read_numbers_binary() 가 읽을 파일은 save_numbers_binary()라는 함수에 의해 만들어졌으며 read_numbers_csv()는 save_numbers_csv()라는 함수에 의해 만들어졌다. 이들 함수의 코드는 주어진 코드에 포함되어 있다. 구현한 함수가 제대로 값들을 읽어 오는 지 간단히 확인하기 위하여 정수 배열의 값 중 가장 큰 값을 구하는 get_max() 함수가 호출된다. 이 함수 역시 소스 코드 내에 포함되어 있다.)

입/출력 예시

␣ : 공백 ↵ : 줄바꿈 ⇥ : 탭

예시 1

입력

```
1
```

출력

```
Max␣:␣2345↵
```