2024 NCKU Program Design I Homework 2

> 請不要嘗試攻擊 Judge 系統，否則此堂課將不予以通過
>
> Don't attack judge system otherwise you will fail this course.
>
> 一旦發現作業抄襲或是請人代寫之情形，學期作業成績將 "全部" 以 0 分計算
>
> One instances of severe plagiarism, hiring someone to write assignments, or similar activities are detected, the semester's assignment scores will be calculated as 0 point across the board.

> 對於作業有任何問題可以直接寄信到 f74114744@gs.ncku.edu.tw (mailto:f74114744@gs.ncku.edu.tw)，如果是其他課程上的問題需要處理請利用全體助教信箱 c2024@mail.csie.ncku.edu.tw (mailto:c2024@mail.csie.ncku.edu.tw)
>
> If you have any question about this homework tasks (ex. problem description), please feel free to contact me (資訊 115 陳俊安, f74114744@gs.ncku.edu.tw (mailto:f74114744@gs.ncku.edu.tw)).
>
> Otherwise, the common problem of this homework please send to TAs mail (c2024@mail.csie.ncku.edu.tw (mailto:c2024@mail.csie.ncku.edu.tw))

Homework 2 Information

Deadline: 10/09 23:59:59

> 不接受遲交
>
> Late submissions are not accepted.

Before you start

- Make sure you can login the server by your personal account.

Goals

- Basic C language Input and Output
- Variable and Type
- Expression
- File I/O

Grading (Total: 100 points)

Task	Score	Tags
A	20	argc & argv
B	20	File I/O
C	20	File I/O
D	20	Formatted Input & Output
E	20	Formatted Input & Output

Submission

- Server IP: `140.116.246.48` , Port: `2024`

```
ssh 學號@140.116.246.48 -p 2024
```

- Login the system by your personal account. (Use the ssh command)
- Create an directory with name "HW2" in your home directory.
  - You can use the "pwd" command to confirm your current directory.
  - The "mkdir [name]" command can create a directory with the name [name]
- In HW2 directory, you need to create 5 files with name "pA.c", "pB.c", "pC.c", "pD.c" and "pE.c" respectively.
- You can directly use the command `hw2` to check whether the result of each question is correct.

> The command is not available at the moment. We will update it as soon as possible.

> 2024/09/26 輸出完後請記得換行，否則在 Server 上會被判定是錯誤的

Tasks

Problem A. Argument count validation (pA.c) - 20%

In the C language, there is a special variable called `argc` , which can be used to count how many arguments we attach when executing the program. Please design a program to calculate this

> It is guaranteed that the number of parameters will not exceed 100.

▶ Hint

Sample Command Testing 1

```
gcc -o pA pA.c
```

```
./pA Apple Banana Guava
```

Sample Output 1

```
3
```

Sample Command Testing 2

```
gcc -o pA pA.c
```

```
./pA OwO
```

Sample Output 2

```
1
```

Problem B. File Input and Output (pB.c) - 20%

Please design a program that reads a file specified by the command-line argument. The program should read four positive integers a, b, c, d from the file, compute the result of $\frac{a}{b} + \frac{c}{d}$, and write the result to the file `answer.txt` .

The format should be `denominator(分子)/numerator(分母)` .

> The final answer does not need to be reduced to its simplest fraction, but you must ensure that both the numerator and denominator do not exceed 10^9.

> Each line of output must end with a newline.

> $1 \leq a, b, c, d \leq 100$

▼ Hint (Write file)

You can try the output of the following program.

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    FILE *fptr;
    fptr = fopen("answer.txt","w"); // write mode
    fprintf(fptr,"Hello world!");
    fclose(fptr);

    return 0;
}
```

▼ Hint (Read file)

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

int main(int argc, char *argv[]) {
    FILE *file;
    file = fopen("testing_input.txt", "r");
    int number;
    fscanf(file, "%d", &number)
    printf("%d\n",number);
    fclose(file);

    return 0;
}

```

Sample Testing

testing_input.txt

```
1 2 3 4
```

Command

```
gcc -o pB pB.c
```

```
./pB testing_input.txt
```

answer.txt

```
5/4
```

Problem C. A simple fraction calculator. (pC.c) - 20%

Please note that this problem continues from Problem B, but the difference is that the input and output file names are no longer limited to `testing_input.txt` and `answer.txt`.

Please design a program where the user is first prompted to input the name of the file to read from, followed by the name of the file to write the result to. The program should then output the computed result to the specified output file.

> Ensure that the file name for the output does not already exist.
>
> If the input file name does not exist, the entire set of inputs must be re-entered.

> Your output format must match the following example exactly; otherwise, the script may judge the output as incorrect.

> If the file name entered is `exit`, the program must terminate.

> As for how to determine whether the input file exists, I'm sure you can easily find it by Google.

> Since everyone hasn't learned about loops yet, the loop part has already been written below. This program allows you to continuously enter strings until you enter 'exit' to leave the loop. You can try running this program yourself to keep entering strings.

```c
#include <stdio.h>
#include <string.h>

int main() {
    char input1[100],input2[100];
    while (1) {
        printf("Please enter the name of the file to read from.\n");
        fgets(input1, sizeof(input1), stdin); // input file name (read)
        input1[strcspn(input, "\n")] = 0;

        if (strcmp(input, "exit") == 0) {
            break;
        }

        if( ??? ) { // the file name of input is not exist
            continue;
        }

        printf("Please enter the name of the file to write the output to.\n");
        fgets(input2, sizeof(input2), stdin); // input file name (write)
        input2[strcspn(input, "\n")] = 0;

        // do something
    }

    return 0;
}

```

Sample Testing

input01.txt

```
1 2 3 4
```

apple.txt

```
1 2 3 4
```

Execution flow: < represents output, > represents input. (Each line of output must end with a newline.)

```
< Please enter the name of the file to read from.
> input01.txt
< Please enter the name of the file to write the output to.
> output01.txt
< Calculation complete!
< Please enter the name of the file to read from.
> Guava.txt
< The file name does not exist.
< Please enter the name of the file to read from.
> apple.txt
< Please enter the name of the file to write the output to.
> banana.txt
< Calculation complete!
< Please enter the name of the file to read from.
> exit
```

output01.txt

```
5/4
```

banana.txt

```
5/4
```

Problem D. Calendar (pD.c) - 20%

Design a program that takes the starting day of the week and the number of days in a month as input, and outputs the calendar for that month.

> From left to right, in order, they are Monday to Sunday, with a single space between any two days, and each day is by default reserved for two spaces. You can refer to the example for detailed output.

> You only need to use the code template below and modify the specified sections accordingly.

> This problem only tests whether you know how to use `printf` to leave two spaces for each number.

```c
#include <stdio.h>

void print_month_calendar(int days_in_month,int start_day) {

    for (int i = 0; i < start_day - 1; i++) {
        printf("   "); // Output spaces until the dates start.
    }

    for (int day = 1; day <= days_in_month; day++) {
        printf("???",???); // Please fill in the correct code here.
        if ( ( day + start_day - 1 ) % 7 == 0) {
            printf("\n"); // This line has already output up to Sunday, print a n
        }
               else {
                      printf(" ");
               }
    }

    printf("\n");
}

int main() {
    int days_in_month,start_day;
    scanf("%d %d",&days_in_month,&start_day);
    // Enter the number of days in the month and the starting day of the week.
        print_month_calendar(days_in_month,start_day);
    return 0;
}

```

Input Limits

- $1 \leq$ `days_in_month` ≤ 12
- $1 \leq$ `start_day` ≤ 7
- Ensure that all inputs are positive integers.

Sample Input 1

```
31 7
```

Sample Output 1

```
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

Sample Input 2

```
28 2
```

Sample Output 2

```
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28
```

Problem E. Text Data Extraction (pE.c) - 20%

Colten has many pieces of text that contains five numbers, with arbitrary characters interspersed between them.

In addition to this, in each case, the text will start with the string "NCKU."

Please design a program to extract these five numbers and output them.

Input Format

```
NCKU[number][character][number][character][number][character][number][character][number
```

Output Format

```
"[number]" "[number]" "[number]" "[number]" "[number]"
```

- The characters will only appear in the following varieties:
  - ^ , % , $, # , a-z , A-Z , ! , :
- $0 \leq number \leq 10^{18}$
- The numbers may have leading zeros, but will not exceed 10 digits.

> The output numbers should not include leading zeros.

> If you are familiar with C language strings, you can use a string-based approach as well, but there is a cleaner and more concise solution available.
>
> If you can't locate this approach, please refer to Hint 2.

▶ Hint 1 (Click to show)

▶ Hint 2 (Click to show)

Sample Input 1

```
NCKU0004c7891a110%3^5001
```

Sample Output 1

```
"4" "7891" "110" "3" "5001"
```

Sample Input 2

```
NCKU01004c7891a110%3^5001
```

Sample Output 2

```
"1004" "7891" "110" "3" "5001"
```

Sample Input 3

```
NCKU1O1u1O116o7
```

Sample Output 3

```
"1" "1" "1" "116" "7"
```

Sample Input 4

```
NCKU2023%09@21!03:00
```

Sample Output 4

```
"2023" "9" "21" "3" "0"
```