Information Processing 2 — Week 2 exercises

Complete as many exercises as you can in-class on your own laptop. You do not need to submit the answers online. Instead, ask an instructor to check your answer when prompted to do so.

> **Note:** You can download the sample programs shown below (questions 1, 5, and 8) and a text file for testing (‘`02-test.txt`’) from Teams.

1 Using symbolic constants in a temperature table

Last week we wrote a program to generate a temperature conversion table. The following version of that program uses a symbolic constant to define the lower limit of the range of temperatures printed. (Download this file from Teams: ‘`02-1-temperature-0.c`’)

```c
#include <stdio.h>

#define LOWER 0 // lower limit of temperature

int main()
{
    float fahr, celsius;
    int upper, step;

    upper = 100; // upper limit of temperature
    step = 10;   // step size between temperatures

    for (celsius = LOWER; celsius <= upper; celsius = celsius + step) {
        fahr = 32 + celsius * 9 / 5;
        printf("%6.1f %6.1f\n", celsius, fahr);
    }

  return 0;
}
```

Modify the program to use two more symbolic constants, `UPPER` and `STEP`, to define the upper limit and step size for the temperatures.

▷▷ Ask an instructor to check and record your work.

2 Numeric values of blank characters

Write a program that prints the numeric values of the blank characters ‘`\t`’ (tab), ‘`\n`’ (newline), and ‘ ’ (space).

▷▷ Ask an instructor to check and record your work.

3 Escaping tabs and backslashes

The two-character sequences ‘`\t`’ for tab and ‘`\n`’ for newline are called ‘escape sequences’. In a string you can also include a backslash literally using another escape sequence ‘`\\`’.

Write a program to ‘escape’ tabs, newlines, and backslashes that occur in the input. For example, the result of running the program on ‘`02-test.txt`’ should be:

```
\n\n\nthis\nis\n\n\na\n\ntest\n\n\n\nfile for  lines   \t\t \tand\tspaces.\n\n\n
```

▷▷ Ask an instructor to check and record your work.

4 Numeric values of characters

Write a program that prints each of the characters between space (`' '`) and tilde (`'~'`) followed by their decimal numeric value. Print one character per line. Use a `for` loop instead of a `while` loop. Hints:

- This program will look like a simpler version of the Celsius/Fahrenheit conversion table program from last week.
- You can use the character constants (`' '` and `'~'`) to set the lower and upper limit of the integer variable that controls the loop. (A character constant such as `'~'` has type `int` and is equal to the value of the integer representing that character.)
- You can use `printf("%c %d\n", c, c)` to print the character `c` followed by its numerical value.

The first and last few lines of your table should look like this:

```
$ ./02-X-characters
  32
! 33
" 34
 ...
| 124
} 125
~ 126
```

What do you notice about the values of the digits (0 to 9) and the letters (`A` to `Z` and `a` to `z`)?

▷▷ Ask an instructor to check and record your work.

5 Counting digit frequencies

The following program counts the frequency of each digit in the input. (Download this file from Teams: '02-5-digitcount-0.c')

```c
#include <stdio.h>

int main()
{
    int c;
    int d[10];

    d[0] = d[1] = d[2] = d[3] = d[4] = d[5] = d[6] = d[7] = d[8] = d[9] = 0;

    while ((c = getchar()) != EOF) {
        if (c == '0') ++d[0];
        if (c == '1') ++d[1];
        if (c == '2') ++d[2];
        if (c == '3') ++d[3];
        if (c == '4') ++d[4];
        if (c == '5') ++d[5];
        if (c == '6') ++d[6];
        if (c == '7') ++d[7];
        if (c == '8') ++d[8];
```

```
        if (c == '9') ++d[9];
    }

    printf("0 %d\n", d[0]);
    printf("1 %d\n", d[1]);
    printf("2 %d\n", d[2]);
    printf("3 %d\n", d[3]);
    printf("4 %d\n", d[4]);
    printf("5 %d\n", d[5]);
    printf("6 %d\n", d[6]);
    printf("7 %d\n", d[7]);
    printf("8 %d\n", d[8]);
    printf("9 %d\n", d[9]);

    return 0;
}
```

Test this program by running it with its own source file as input. (The result should be eight '0's, six '1's, and five of every other digit.)

Improve the program in the following ways:

- An array can be initialised in its declaration with a brace expression, like this:

  ```
  int x[3] = { 1, 2, 3 };
  ```

  In the above program, initialise the array `d` in its declaration using a brace expression. (The explicit initialisations of each element on the next line can be removed.)
- Because characters are integers you can perform arithmetic on them and compare them like integers. For example, to test for a digit you can use '`if ('0' <= c && c <= '9')`'. You can also calculate the correct array index for a digit using the expression '`d[c-'0']`'. Replace the ten `if` statements with a single statement that checks if the character is a digit and then increments the correct array location by calculating the corresponding index.
- Replace the ten `printf` function calls with a loop that calls `printf` ten times to print the ten digit frequencies.

Hint: the final program should only have about 10 lines of code in the body of the main function.

▷▷ Ask an instructor to check and record your work.

6 Print a histogram

Modify the program from the previous exercise so that for each digit it prints the digit, the frequency count for the digit, and a histogram made by calling '`putchar('*')`' the same number of times as the frequency of the digit. For example, running the modified program on its own source file gives the following result for me:

```
0  16 ****************
1   1 *
2   0
3   1 *
4   0
5   0
```

```
6    0
7    0
8    0
9    2 **
```

▷▷ Ask an instructor to check and record your work.

7 More frequency counting

Modify your program that counts the frequencies of digits so that it counts and prints the frequencies of all characters. Include all blank characters (tab, newline, space). When printing the results, ignore (do not print) zero frequencies (characters that do not appear at all in the input).

Hints:

- Only count characters in the range 0 to 126 (inclusive).
- The frequency array will have to be large enough to count 127 different characters with numeric values in the range 0 to 126.
- The frequency array will be too large to initialise using a brace expression. Instead, fill it with zeros using a loop before you begin to read the input.
- When printing the results, treat tab and newline specially so that you can print them using their escape sequences '`\t`' and '`\n`'.
- For other special characters with numeric values less than ' ' (space), you can print them using the format `"\\%03o"` (an escaped backslash then percent, zero, three, and the lower case letter '`o`'). This is the traditional way to represent these special characters in string and character constants.

For example, running my sample solution on the file '`02-test.txt`' produces this output:

```
\t      4
\n     16
        7
.       1
a       3
c       1
d       1
e       4
f       2
h       1
i       4
l       2
n       2
o       1
p       1
r       1
s       6
t       3
```

▷▷ Ask an instructor to check and record your work.

8 Counting words

The following program detects the beginning of words and prints the initial character in each word that it detects. (Download this file from Teams: '02-8-words-0.c')

```
#include <stdio.h>

#define OUTSIDE  0  // not in a word
#define INSIDE   1  // in a word

int main()
{
    int c, state;

    state = OUTSIDE;

    while ((c = getchar()) != EOF) {
        if (c == ' ' || c == '\t' || c == '\n') {
            state = OUTSIDE;
        }
        else {
            if (state == OUTSIDE) {
                printf("word begins with %c\n", c);
                state = INSIDE;
            }
        }
    }

    return 0;
}
```

Compile this program and test it using the file '02-test.txt' which is available via Teams. (Place the '02-test.txt' file in the same directory as the program. You can then run the program while reading input from the file '02-test.txt' like this:

```
$ ./02-8-words-0 < 02-test.txt
word begins with t
word begins with i
word begins with a
word begins with t
word begins with f
word begins with f
word begins with l
word begins with a
word begins with s
```

If you cannot make this work, ask for help.)

Modify the above program so that it *counts* the number of words in the file and then *prints* that number at the end of execution. Test it using the same input file '02-test.txt'.

Hints: All you have to do it declare an integer variable to count the number of words, initialise it to 0, add 1 to it each time you find the beginning of a word, and print it out at the end of the program.

▷▷ Ask an instructor to check and record your work.

9 Counting characters, words, and lines

Extend the program from the previous question so that it also counts the number of characters and the number of lines in the file. Print all three totals at the end of the program. Test your program using the file '`02-test.txt`' which has 60 characters, 9 words, 16 lines.

▷▷ Ask an instructor to check and record your work.

10 Removing consecutive spaces

Write a program that copies input to output but replaces multiple consecutive newline characters with a single newline character. Test your program using '`02-test.txt`'.

When it works, modify your program so that it replaces multiple consecutive spaces or tabs (mixed together in any order) with a single space character.

▷▷ Ask an instructor to check and record your work.

11 Challenge (bonus point)

Modify your program from exercise 4 so that it prints a neat table of integers with their corresponding character, like this:

```
 32      33 !    34 "    35 #    36 $    37 %    38 &    39 '
 40 (    41 )    42 *    43 +    44 ,    45 -    46 .    47 /
 48 0    49 1    50 2    51 3    52 4    53 5    54 6    55 7
 56 8    57 9    58 :    59 ;    60 <    61 =    62 >    63 ?
 64 @    65 A    66 B    67 C    68 D    69 E    70 F    71 G
 72 H    73 I    74 J    75 K    76 L    77 M    78 N    79 O
 80 P    81 Q    82 R    83 S    84 T    85 U    86 V    87 W
 88 X    89 Y    90 Z    91 [    92 \    93 ]    94 ^    95 _
 96 `    97 a    98 b    99 c   100 d   101 e   102 f   103 g
104 h   105 i   106 j   107 k   108 l   109 m   110 n   111 o
112 p   113 q   114 r   115 s   116 t   117 u   118 v   119 w
120 x   121 y   122 z   123 {   124 |   125 }   126 ~
```

▷▷ Ask an instructor to check and record your work.