

Coding Review (I2P 2019)

Standard Outputs of C v1.0

Review of common inputs: `printf`, `putchar`, `puts`.

Comparing to Standard Inputs, Standard Outputs are much simpler.

Basic Formats

1. What is the result of the following code and input?

```
#include <stdio.h>

int main(void) {
    char a[5] = {'A', 'B', '\\0', 'C', 'D'};
    puts(a);
    return 0;
}
```

Answer: (Write your answer here!)

2. Replace `<REPLACE_HERE>` with a string format that can get the expected output.

```
#include <stdio.h>

int main(void) {
    printf("<REPLACE_HERE>");
    return 0;
}
```

Expected Output:

```
printf("%d\\n", x);
```

Answer: (Write your answer here!)

3. What is the result of the following code and input?

```
#include <stdio.h>

int main(void) {
    int a, b;
    a = 101;
}
```

```

    b = 8787887;
    printf("%8d\n", a);
    printf("%8d\n", b);
    printf("%08d\n", a);
    printf("%08d", b);
    return 0;
}

```

Answer: (Write your answer here!)

4. What is the result of the following code and input?

```

#include <stdio.h>

int main(void) {
    float f;
    f = 878722e-4;
    printf("%f\n", f);
    printf("%.2f\n", f);
    printf("%.1f\n", f);
    printf("%.0f\n", f);
    printf("%.f", f);
    return 0;
}

```

Answer: (Write your answer here!)

5. Replace `<REPLACE_HERE>` with a string formats that can get the expected output.

```

#include <stdio.h>

int main(void) {
    long long x, y;
    scanf("%lld%lld", &x, &y);
    printf("<REPLACE_HERE>\n", 20, 2*(unsigned long long)x);
    printf("<REPLACE_HERE>\n", 20, 2*(unsigned long long)y);
    return 0;
}

```

Input:

```

9100000000000000000
12

```

Expected Output:

```
18200000000000000000
00000000000000000024
```

Answer: (Write your answer here!)

6. What is the result of the following code and input?

```
#include <stdio.h>

int main(void) {
    putchar('\a');
    return 0;
}
```

Answer: (Write your answer here!)

Standard I/O Review

1. `scanf("%c", ...)`, `getchar` does not ignore leading whitespace characters.
2. `gets` does not store the terminating newline character; `fgets` stores the terminating newline character (if the input is terminated by newline instead of `EOF`).
3. when reading `EOF`, `scanf`, `getchar` returns `EOF`; `gets`, `fgets` return `NULL`.
4. When reading strings, remember to save an additional space for the easily forgotten `'\0'`.
5. Strings should be null-terminated (end with `'\0'`) before outputting using `printf("%s", ...)` or `puts`.

The list above are some mistakes that I see a lot of beginners make. If you see other special usages, you can search for them online. (such as `%x`, `%#x`, `%hd`, ...)

If you forget some of the I/O formats above in your exam (such as leading zero paddings), most of them can be replaced with additional `if` statements and loops.

For the next assignment, we'll review some basic syntaxes of C.

Epilogue

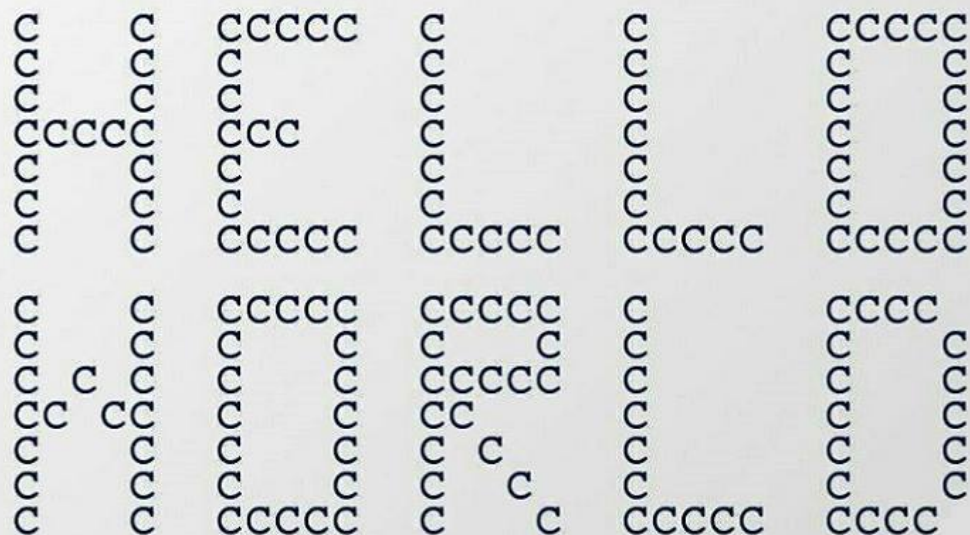
Me:

I am good in C language.

Interviewer:

Then write "Hello World" using C.

Me:



The image shows the words "HELLO" and "WORLD" in a pixelated font. Each character is constructed from multiple 'C' characters. The 'H' is formed by two vertical lines of 'C's and a horizontal line connecting them. The 'E' is formed by a vertical line and three horizontal lines. The 'L' is formed by a vertical line and a horizontal line at the bottom. The 'O' is formed by a square outline of 'C's. The 'W' is formed by two 'V' shapes side-by-side, each made of 'C's. The 'R' is formed by a vertical line, a horizontal line, and a curved line at the bottom. The 'D' is formed by a vertical line and a semi-circular shape made of 'C's. The 'O' is formed by a square outline of 'C's.

Photo Credit: Posted on [Reddit](#)

If there's any typo, please discuss on iLMS or email j3soon@gapp.nthu.edu.tw, I appreciate your help.