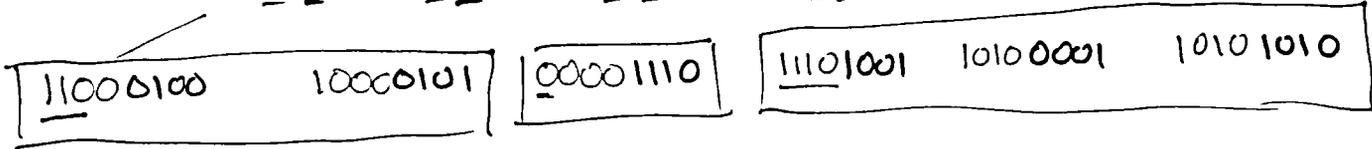


0xC4850EE9A1AA

(81)

Q1: How many UTF-8 code points are encoded by:

0xC4 0x85 0x0E 0xE9 0xA1 0xAA



Q2: ~~What~~ Explain in 1 sentence what uppercase-ascii does (on handout)

Takes a C string and changes the string to replace <sup>all</sup> lowercase ASCII characters with their uppercase equivalent

Q3: Explain in 1 sentence what truncate-to-n does (on handout)

Takes a C string s and a length n and ~~the~~ changes s by setting the nth index to a null terminator.

```

1 #include <string.h>
2 #include <stdio.h>
3 #include <stdint.h>
4
5
6
7
8 void uppercase_ascii(char s[]) {
9     for(int i = 0; s[i] != '\0'; i += 1) {
10        if(s[i] >= 'a' && s[i] <= 'z') { s[i] -= 32; }
11    }
12 }
13
14
15
16
17
18 void truncate_to_n(char s[], uint32_t n) {
19     s[n] = '\0';
20 }
21
22 int main() {
23     char input[100];
24     fgets(input, 100, stdin);
25     printf("The input was: %s\n", input);
26     truncate_to_n(input, 6);
27     printf("First 6 bytes of input: %s\n", input);
28     uppercase_ascii(input);
29     printf("Input with ASCII uppercased: %s\n", input);
30 }

```

*this will stop  
 the loop at  
 1st \0*

**Problem Statement:**

Write a program that takes an input string and prints the first 6 bytes of that string and that string with all of its lowercase ASCII characters uppercased.

**What should happen according to the problem statement?**

```

$ gcc analyzer.c -o analyzer
$ ./analyzer
utf8 is ☹️
The input was: utf8 is ☹️

First 6 bytes of input: utf8 i
Uppercased: UTF8 IS ☹️

```

**What does happen according to this implementation?**

```

$ gcc analyzer.c -o analyzer
$ ./analyzer
utf8 is ☹️
The input was: utf8 is ☹️

First 6 bytes of input: utf8 i
Uppercased: UTF8 I

```

