```c
int8_t is_ascii_lessthan(char c) {
  return c <= 128;
}
```

```c
int8_t is_ascii_bitand(char c) {
  return (c & 0x10000000) == 0;
}
```

Consider each of these inputs. What does each produce on each of the implementations? Why?

- `0x48 (0b01001000)`
- `0xFF (0b11111111)`
- `0x0A (0b00001010)`

What is the code point encoded by each of the following UTF-8 byte sequences? You may find it useful to give your answer as the hexademical value of the code point. (e.g. code point 200 would be `0xC8`)

- `0xE2 0x9C 0x94` (heavy check mark ✔)
- `0xC3 0xAA` (e with circumflex, ê)

What is the UTF-8 encoding of the following code points?

- `0x3bb` (lambda, λ)
- `0x1F600` (grinning face, 😀)

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

int32_t code_point2(char str[]) {
    char c1 = str[0], c2 = str[1];
    return ((c1 & 0b00011111) << 6) + (c2 & 0b00111111);
}

int32_t code_point3(char str[]) {




}

int32_t code_point4(char str[]) {
    char c1 = str[0], c2 = str[1], c3 = str[2], c4 = str[3];
    return ((c1 & 0b00001111) << 18) + ((c2 & 0b00111111) << 12) + ((c3 & 0b00111111) << 6) + (c4 & 0b00111111);
}

int32_t codepoint_of(char str[]) {
    if ((str[0] & 0b10000000) == 0) { return str[0]; }
    else if ((str[0] & 0b11100000) == 0b11000000) { return code_point2(str); }
    else if ((str[0] & 0b11110000) == 0b11100000) { return code_point3(str); }
    else { return code_point4(str); }
}

int main() {
    char checkmark[] = "✔";  // same as {0xE2, 0x9C, 0x94, 0x00}
    int32_t cp = codepoint_of(checkmark);
    printf("Code point: %d 0x%X\n", cp, cp);
    char e_hat[] = "ê";      // same as {0xC3, 0xAA, 0x00}
    int32_t cp2 = codepoint_of(e_hat);
    printf("Code point: %d 0x%X\n", cp2, cp2);
}
```