**C program that converts a positive integer (1–3999) to its Roman numeral representation. The program uses a greedy method with known Roman numeral values and symbols.**

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

void intToRoman(int num, char *res) {
   // Roman numeral mappings (values paired with symbols)
   int vals[]  = {1000, 900, 500, 400, 100, 90,  50, 40,  10, 9,   5,  4,   1};
   char *syms[] = {"M", "CM","D", "CD","C","XC","L","XL","X","IX","V","IV","I"};
   res[0] = '\0'; // start with empty string
   for (int i = 0; i < 13; ++i) {
      while (num >= vals[i]) {
         strcat(res, syms[i]);
         num -= vals[i];
      }
   }
}

int main(void) {
   int n;
   char roman[128]; // big enough buffer for roman numeral output
   printf("Enter an integer (1-3999): ");
   if (scanf("%d", &n) != 1) {
      printf("Invalid input.\n");
      return 1;
   }
   if (n < 1 || n > 3999) {
      printf("Number out of range. Roman numerals supported: 1 to 3999.\n");
      return 1;
   }
```

```
    intToRoman(n, roman);
    printf("Roman representation: %s\n", roman);
    return 0;
}
```

**How it works**

- Arrays vals and syms list the integer values and their Roman symbols in descending order (including subtractive forms like 900→"CM", 4→"IV").
- The code repeatedly subtracts the largest possible Roman value from the number and appends the corresponding symbol, until the number becomes 0.
- This greedy approach produces the standard Roman numeral form for values 1 through 3999.

**Examples**

- Input 1994 → Output MCMXCIV
  Steps: 1000 → M (remain 994), 900 → CM (remain 94), 90 → XC (remain 4), 4 → IV → MCMXCIV.
- Input 58 → Output LVIII (50 L, 5 V, 3× I).

**Sample I/O**

Enter an integer (1-3999): 8

Roman representation: VIII

Enter an integer (1-3999): 400

Roman representation: CD

Enter an integer (1-3999): 4000

Number out of range. Roman numerals supported: 1 to 3999.