

# Tutorial 6: Functions

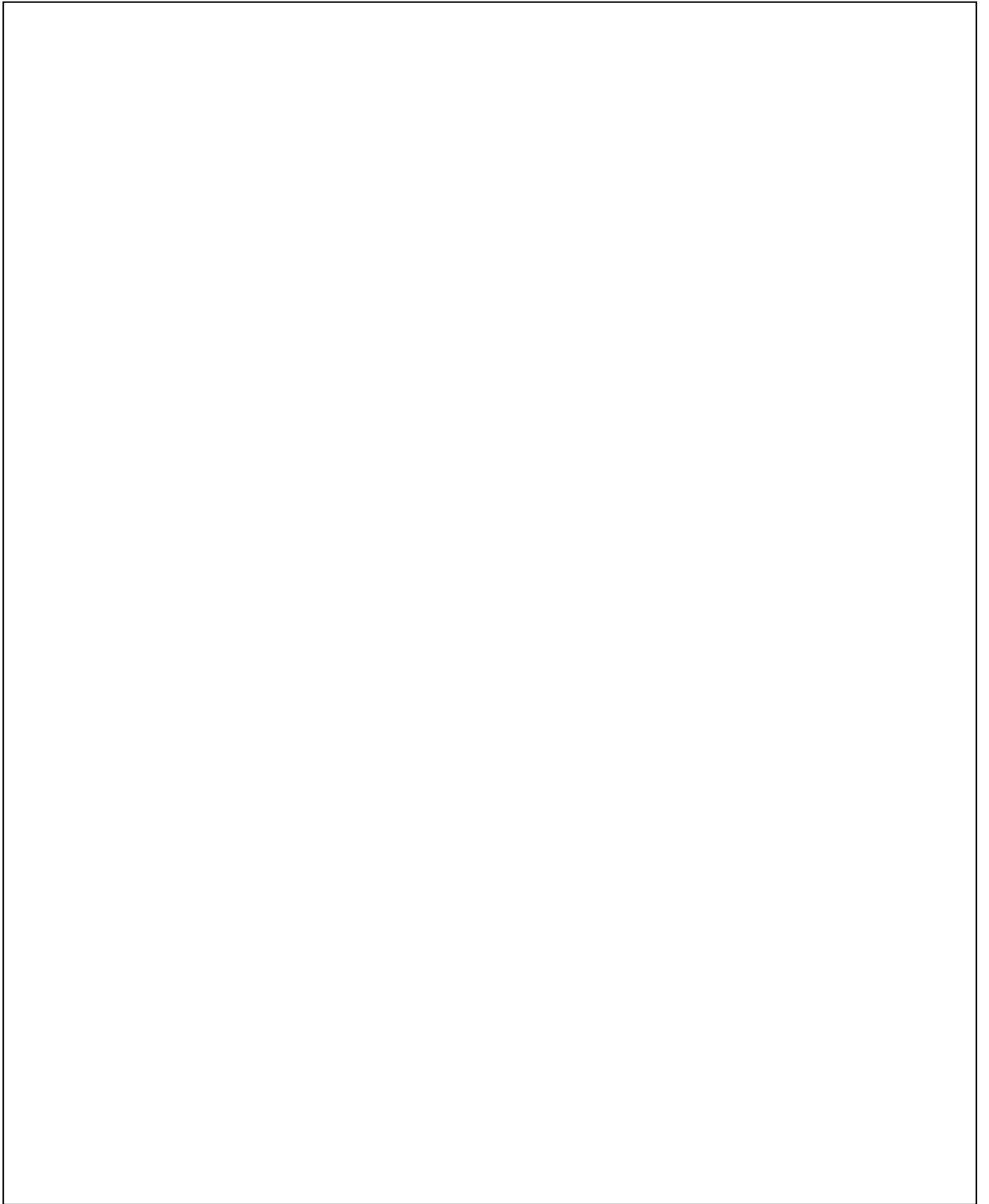
## Problem 1: 40-bit Addition

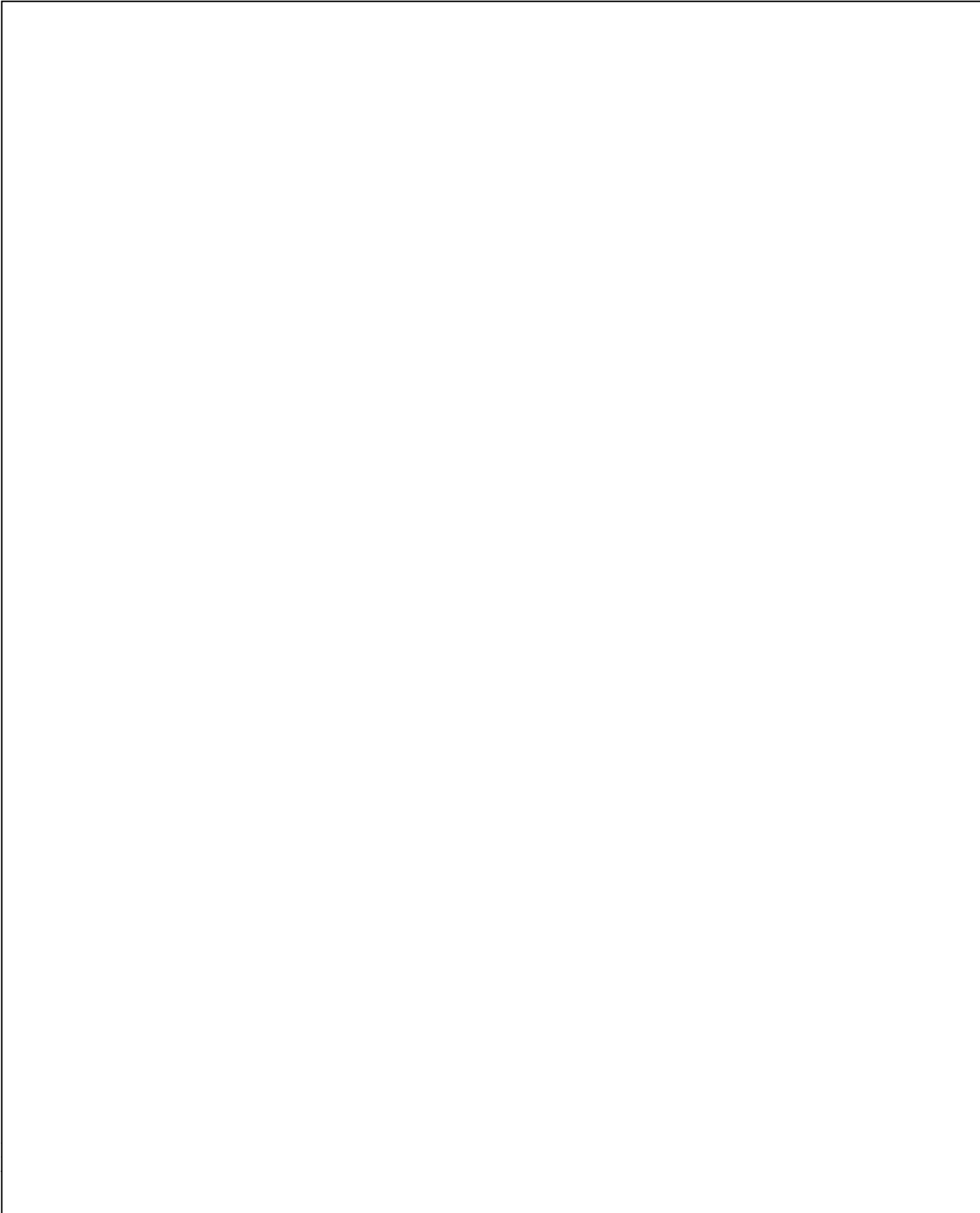
We like to design a function that does addition in 40 bits. First try the following 40-bit additions by hand. Next design this function by developing a C code for it. Modify this function to do addition as well as subtraction.

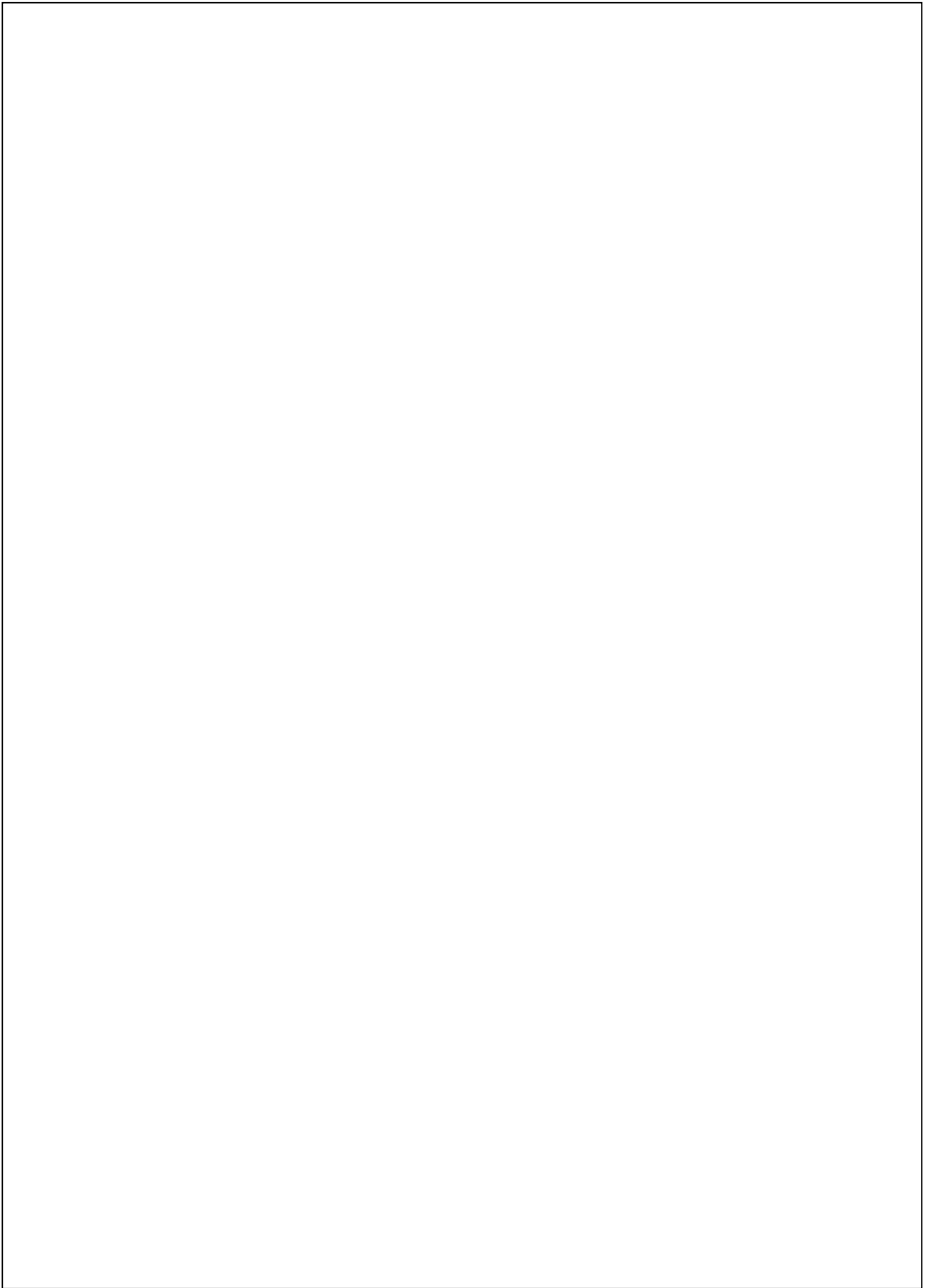
$$0x08\ 0000\ 0020 + 0x02\ 0000\ 0040 =$$

$$0x08\ 0000\ 0020 + 0xf7\ 0000\ 0040 =$$

$$0x08\ 0000\ 0020 + 0x07\ ffff\ ffff = 0x1$$



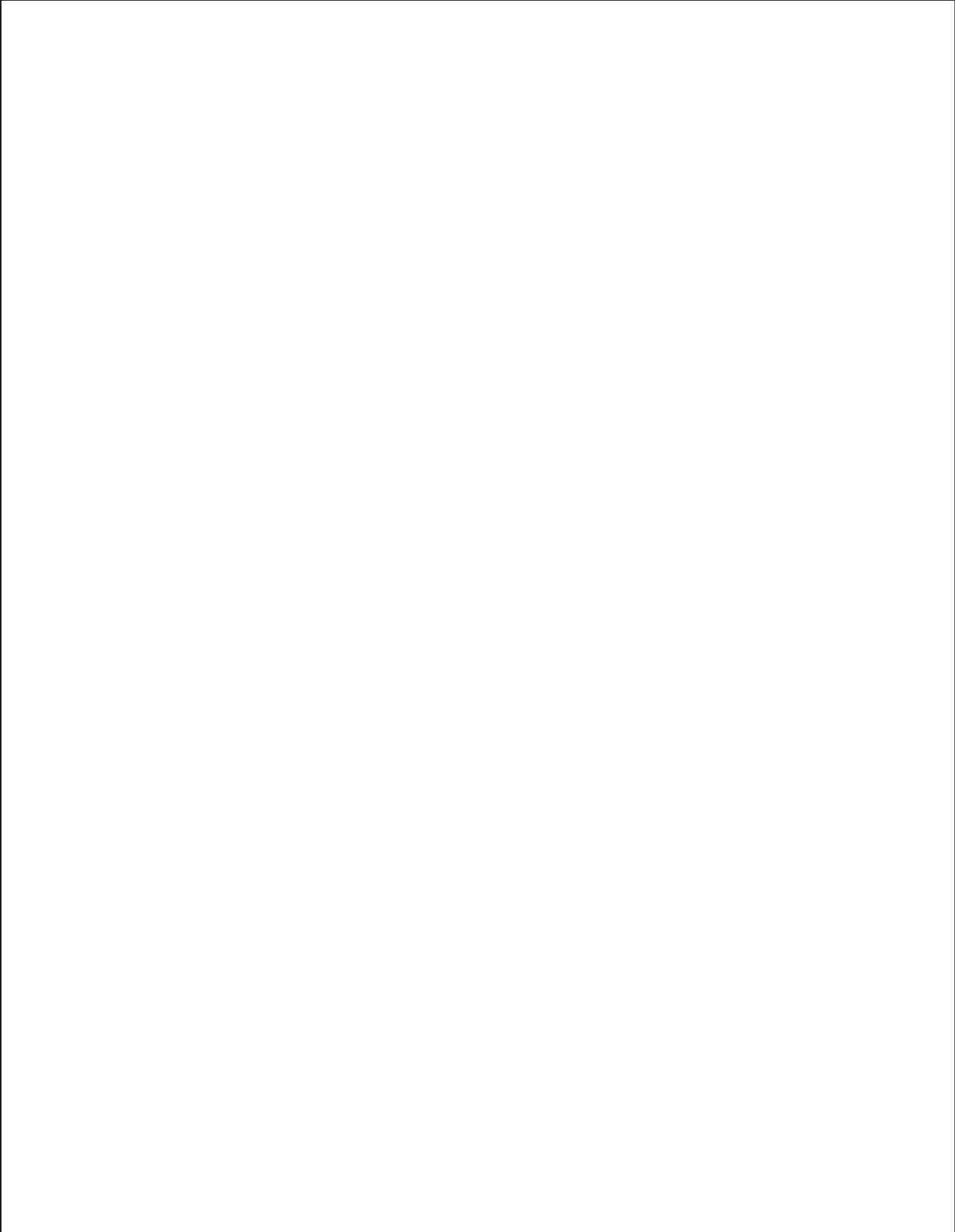




## Problem 2: Reverse Polish Calculator

Write an assembly program for calculation in Reverse Polish Notations.

In reverse polish calculators the operators are post fix. For example the infix expression  $(3 + 8) * 5 - (2 + 6) * 4$  is represented in post fix reverse polish notations as:  $3 8 + 5 * 2 6 + 4 * -$ . In post fix notation the operators follow the operands. Also there is no need for parentheses.



**Problem 3: Parameter Passing to Function**

Consider the C code in Figure 6 and the printout from it in Figure 7.

Explain if you find anything unusual with the printouts in Figure 7. Specifically look at correspondence between the addresses and values of the array elements printed out in Figure 7.

```
#include <stdio.h>

void array_check(int a, int b, int c, int d)
{
    printf("Address      Value \n",&a, a, &b, b, &c, c, &d, d);
    printf("%x      %x\n%x      %x\n%x      %x\n",&a, a, &b, b,
    &c, c, &d, d);
}

int main(void)
{
    int arr [] = {1,2,3,4,5,6};

    array_check(arr[0], arr[1], arr[2], arr[3]);
    array_check(arr[3], arr[2], arr[1], arr[0]);

    return 0;
}
```

**Figure 6: Passing Parameters in C-Program**

Address	Value
bffff270	1
bffff274	2
bffff278	3
bffff27c	4
Address	Value
bffff270	4
bffff274	3
bffff278	2
bffff27c	1

**Figure 7: The Outputs of printf Statements**

