**CSE142** **2. Nov. 2000**

**Name:**

**Section:**

# Solutions to Quiz Problems

```
int quizFunction(int x, int *y) {
  x = x + 5;
  *y = *y + 3;
  return x + *y;
}

int main(void) {
  int p, q, r;
  p = 2;
  q = 4;
  r = quizFunction(p, &q);
  /* Break */
  return 0;
}
```

1. What is the value of ‘p’ at the point marked `/* Break */`?

a) 2 b) 4 c) 5 d) 7 e) 14

2. What is the value of ‘q’ at the point marked `/* Break */`?

a) 2 b) 4 c) 5 d) 7 e) 14

3. What is the value of ‘r’ at the point marked `/* Break */`?

a) 4 b) 5 c) 7 d) 9 e) 14

# Solutions to Practice Problems

Today's section deals with an on-going example. Your goal is to construct a simple fractional calculator (this material is based on the examples in Chapter 6 of the textbook). The user can enter 2 fractions, separated by an operation. For example:

```
>>1/2 + 4/5
= 13/10
```

The program reads 2 fractions and an operator from the user (1/2 + 4/5 in the above example). The program then calculates the mathematical result, and displays it after an equal sign (= 13/10 in the above example).

4. The first step is to construct a top-down design of a solution to this problem. The goal is to create a collection of functions that you will need to solve the problem.

4a. What functions are needed in main?

We can assume that main will consist of a loop with 3 steps: read_input, compute_result and output_result. These can be expanded into the following requirements:

read_input

compute_result

output_result

4b. Which of these steps is the most complex? Break this step into logical sub-problems?

There are 4 possible computations needed to compute a result: add, subtract, multiply and divide.

4c. Code re-use is always good. Chances are that some of the steps you indicated in part 4b could be written with a single line of code. How?

The process of subtracting one fraction from another is the same as adding those fractions, once you multiply the second numerator by –1:
`subtract(num1, den1, num1, den2)` gives the same result as
`add(num1, den1, -num1, den2)`

Always be on the look-out for ways to re-use code!

4d. For further practice, try to identify the steps needed to output the result in simplest form. (Simplest form means that the greatest common divisor of the numerator and denominator is 1. I.e., you need to divide the numerator and denominator by their GCD.) You do not need to write any code, just sketch out what functions you would need.

5. Now to write some code. The following will test your knowledge of pointers.

5a. What is the function prototype for add_fraction? This function takes 2 fractions and computes their sum. (You will need output parameters.) Fractions are represented using two ints (one for the numerator and one for the denominator).

```
void add_fraction(int num1, int den1, int num2, int den2,
                  int *result_num, int *result_den);
```

5b. Complete the following code snippet using add_fraction.

```
int num1, num2, num3, den1, den2, den3;
/* Assume num1, num2, den1, den2 have been initialized. */
if (op == '+')
{
  add_fraction(num1, den1, num2, den2, &num3, &den3);
}
```

5c. The function prototype for subtract_fraction will look identical to add_fraction. Write the complete definition of subtract_fraction.

```
void subtract_fraction(int num1, int den1,
                       int num2, int den2,
                       int *result_num, int *result_den)
{
  add_fraction(num1, den1, -num2, den2,
          result_num, result_den);
}
```

Note that since result_num is of type (pointer to int) and add_fraction expects a value of type (pointer to int), you do not need the & operator in front of result_num. The compiler will get very cranky if you try to use an &!

5d. Is the following code correct? If not, does it contain syntax or semantic errors? If it is correct, can you eliminate the local variables?

```
void scan_fraction(int *n, int *d)
{
  int num, den;
  char slash;
  scanf("%d %c %d", &num, &slash, &den);
  *n = num;
  *d = den;
}
```

The code is correct, you can replace the body of the function with:

```
  char slash;
  scanf("%d %c %d", n, &slash, d);
```

5e. Check out the on-line solution of this problem.