CSE 305 Spring 2015

Assignment #4:

Parameter Passing and Python's List Comprehensions

Assigned: April 2, 2015

(amendments in red posted on April 4)

Due: Mon, April 13, 2015 (11:59 pm)

Note: This assignment may be done by a pair of students.

1. Consider the following incorrect C program to compute the normal form of a rational number:

```
typedef struct {
     int numr;
     int denr;
     } RATIONAL;

int main() {

    RATIONAL r;

    int normalize(RATIONAL r) {
        int gcd(int x, int y) {
            while (x != y)
                if (x > y)
                        x = x-y;
                else y=y-x;
            return x;
        }
        int g = gcd(r.numr, r.denr);
        r.numr = r.numr/g;
        r.denr = r.denr/g;
    }

    r.numr = 77;
    r.denr = 88;
    normalize(r);

    printf("%d/%d\n", r.numr, r.denr);
}
```

   a. Why does `normalize` not compute the normal form of the rational number 77/88?

   b. Using call-by-reference for parameter r, show a corrected version of `normalize` along with a suitable call on `normalize` so that the output is the correct normal form of 77/88.

   Name your file **rational2.c** and submit it online. Include the answer for part (a) also in the file as a comment at the top of the file.

2. Consider the following C function for carrying out the summation expressed by the operator $\sum$:

```
int sigma(int *k, int low, int high, int expr()) {
    int sum = 0;
    for (*k=low; *k<=high; (*k)++) {
        sum = sum + expr();
    }
    return sum;
}
```

a. Write a `main` program that makes repeated use of `sigma` in order to compute and print out the value of (parentheses added to clarify grouping) :

$$\sum_{i=0}^{4} \left(i * \sum_{j=0}^{4} \left(j * \sum_{k=0}^{4} k \right) \right)$$

Name your file **sigma.c** and submit it online.

b. Consider the multiplication of two matrices, c = a x b, defined as:

$$\underset{i=0}{\overset{n}{\forall}} i \;\; \underset{j=0}{\overset{n}{\forall}} j \;\; c[i, j] = \sum_{k=0}^{n} a[i, k] * b[k, j]$$

(In original assignment, it was a[i,j]*b[j,k]. This typo has been corrected above.)
Define in C a procedure `matmult` that incorporates the above definition: translate the universal quantifier using a for-loop and the operator $\sum$ using `sigma`. Test `matmult` with:

$$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Use the following `main` procedure to test out your `matmult` definition. It suffices for matmult to work with 2x2 matrices.

```
int main(){
        int a[2][2] = {{1,2}, {3,4}};
        int b[2][2] = {{5,6}, {7,8}};
        int c[2][2];

        matmult(a, b, c);
        printf("%d %d\n%d %d\n",
                c[0][0], c[0][1], c[1][0], c[1][1]);
}
```

Name your file **matrix.c** and submit it online.

3. Consider the Python definition for 'quicksort' using list comprehensions:

```
def qsort(a):
     if a == []: return []
     left  = [x for x in a[1:] if x < a[0]]
     right = [x for x in a[1:] if x >= a[0]]
     return qsort(left) + [a[0]] + qsort(right)

print(qsort([5,4,6,3,7,9,2,8,1,0]))
```

a. i. Replace the + operations in the above code by Python's built-in append/extend operations.

   ii. Replace the list comprehensions by equivalent code making use of Python list generators. Use append/extend in your generated code.

   Test your program to make sure it sorts correctly. Name the file **qsort1.py**, and submit it online.

b. Replace the list generators in **qsort1.py** by equivalent code making use of higher-order functions and thunks.

   Test your program to make sure it sorts correctly. Name the file **qsort2.py**, and submit it online.

Team-work and On-line Submission:

1. This assignment may be done by a team of at most two students. Write both student names at the top of all program files when submitting them.

2. It is fine to do the assignment solo. Write your name at the top of each file and submit it.

End of Assignment #4