Classical Programming Problems

D. E. Stevenson
442 R. C. Edwards Hall
Department of Computer Science
Clemson University
PO Box 341906
Clemson, SC 29634-1906
Email: steve@cs.clemson.edu

October 30, 1997

1 Brief Description

This file contains "classical" programming problems that one might expect to see in computer science texts. They are meant to be programming exercises for lower level classes; *i. e.*, CS I–IV. However, they can also be used for other classes such as software engineering classes dealing with specifications or program proofs.

Font rules: normal English and mathematical fonts are used as would naturally occur. Programming "words" and program segments are set in `typewrite font`. Concepts that should be explained (but really not defined) are given SMALL CAPITAL AND UNDERLINED; such items are defined in the glossary.

2 Problems Mostly at the College Algebra Level or less

2.1 *Hello, World.* Write a program that writes "Hello, World" to the SCREEN.

2.2 Produce a table of Celcius and the equivalent Fahrenheit tempertures from 0°C to 100°C.[3]

2.3 Write a program that reads in an integer in base ten and writes out two numbers: the original number and then original number represented in base 2.[3]

2.4 Develop an algorithm to give exact change for a purchase. The input is the purchase (in dollars and cents) and the payment value. The purchase is in decimal notation. The output is the number of U. S. coins and bills. Assume that the coins are the penny, nickel, dime, and quarter. The bills are the $1, $5, $10, and $20.[3]

2.5 Write a program to sort exactly three numbers using only `if-then-else` statements.[3]

2.6 Write a program that reads in three real numbers. Determine computationally which of the following cases are true: the three numbers

(a) Do not represent a triangle.
(b) Represent an equilateral triangle.
(c) Represent an isosceles triangle.
(d) Represent a right triangle.
(e) Represent a triangle that is not one of the three immediately above cases.

[3]

2.7 *Day of Week.* A program for converting Gregorian dates in the form *month-day-year* to *day of the week* is the following. Let the Gregorian date be represented by $M/D/Y$ where we have the following definitions:

- M is the month of the year. Let m be the month number derived from M by the rule m is $M - 2$ if $M \geq 3$ and m is $M + 10$ otherwise.
- d is the day of the month.
- y is the year of the century.
- c is the number of the *previous* century.

The algorithm is as follows:

(a) A is the integer part of $(13m - 1)/5$.
(b) B is the integer parg of $y/4$.
(c) C is the integer part of $c/4$.
(d) $D = A + B + C + d + y - 2c$.
(e) Make R equal to the remainder of $D/7$.
(f) Interpret R as Sunday if $R = 0$, Monday if R is 1, *etc.*

[3]

2.8 *Julian Day.* The Julian Day for any day A. D. can be computed by the below formula:

$$Julian = 1721060 + 365y + 31(m - 1) + d + \lfloor y'/4 \rfloor - \lfloor y'/100 \rfloor + \lfloor y'/400 \rfloor - x$$

where $\lfloor z \rfloor$ is the largest integer less than or equal to z, y is the year, m is the month, d is the day. For $m \leq 2$, $x = 0$ and $y' = y - 1$. For $m > 2$, $x = \lfloor .4m + 2.3 \rfloor$ and $y' = y$.[3]

2.9 Write a program to reverse the digits of an integer number stored in the computers memory and write the number and its reverse to the screen. [3]

2.10 *Check Digits.* Credit cards usually have a so-called *check digit.* This is a single digit that is assigned when the account number is developed and has a special property. One particularly simple mechanism is to assign the last digit of the sum of all the other digits. For example, suppose we have a nine-digit account number (including the check digit). The check digit would be the sum of the eight digits. This digit could be placed anywheres in the sequence, say the third digit. As a full example, suppose the eight numbers are 12345678. Their sum is 36; thus, 6 is the check digit. The account number is therefore 126345678.

Write a program to read in a nine digit integer from the keyboard and check it for these rules. Write "okay" or "not okay" for the results of the test.[3]

2.11 *Cryptoarithmetic.* In cryptoarithmetic problems, we are given a problem wherein the digits are replaced with characters representing digits. A *solution* to such a problem is a set of digits that, when substituted in the problem, gives a true numerical interpretation. Example:

$$\begin{array}{r} IS \\ IT \\ \hline OK \end{array}$$

has a solution $\{I = 1, K = 1, O = 3, S = 5, T = 6\}$. For each of the below cryptoarithmetic problems, write a program that finds **all** the solutions **in the shortest possible time.**

$$\begin{array}{r} IS \\ IT \\ \hline OK \end{array} \quad \begin{array}{r} I \\ AM \\ \hline OK \end{array}$$

[3]

2.12 *Farmer's Dilemma.* A farmer is taking a sack of corn, a goose, and a fox to market. The farmer approaches the south bank of a river and must cross the river to get to the market. There is a boat with just enough room for the farmer and one of the commodities. Devise a strategy for the farmer to get the three commodities across the river without losing one.

3 Simple Number Theory Problems

3.1 Write a program that computes the arithmetic sum of a constant difference sequence. I. e.:

$$\begin{aligned} d_0 &= a \\ d_{n+1} &= cd_n + b \end{aligned}$$

Run the following values and compare to the closed form solution. Maximum index = 10,000, $a = 1.0, c = 5, b = 2^{-15}$.

3.2 Write a program that computes the sum of a geometric series.

$$\begin{aligned} d_0 &= a \\ d_{n+1} &= d_n + c^n \end{aligned}$$

Run the following values and compare to the closed form solution. Maximum index = 10,000, $a = 1.0, c = 5, b = 2^{-15}$.

3.3 Tabulate the function $k = 2^n$ for $n = 1, \ldots, 50$. Do this for the *fewest possible multiplications*.[3]

3.4 It is conjectured that for any $n > 0$, $n^2 + 3n + 5$ is never divisible by 121. Test this conjecture for $n = 1, 2, \ldots, 10000$.[3]

3.5 Code the Fibonacci sequence = $\{1, 1, 2, 3, 5, 8 \ldots\}$.[3]

3.6 Compute the factorial defined by the rule

$$n! = n \times (n-1) \times \ldots \times 2 \times 2 \times 1.$$

Test your routine for values of $n = 1, 2, 3, \ldots, 50$. What do you observe?[3]

3.7 Write a program to produce a table showing that the following equality is true for $k = 0, \ldots, 49$:

$$(k+1)^2 = k^2 + (2k+1).$$

[3]

3.8 *Russian Peasant Multiplication.* This program allows one to do multiplication by just doing muliplication or division by 2 (and addition). This is interesting since computers generally use base 2 arithmetic. The algorithm is as follows. Start two rows, one with the multiplicand (say, A) and the other the multiplier (say B). The goal is to multiply A by B. The first number is successively divided by 2 while the second number is multiplied by 2. If the number in the first column is odd, then the correspoinding number in the second column is added to the sum. The final sum is the product. Example:

Multiplicand	Multiplier	Product
37	41	41+0 = 41
18	82	
9	164	164+41 = 205
4	328	
2	656	
1	1312	1312+205 = 1517

[3]

3.9 Write a program to find all the factors of a non-negative integer.[3]

3.10 Write a program to determine if a given input non-negative integer is prime.[3]

3.11 *Sieve of Eratosthenes.* Using the Sieve principle, write a program that can produce all the prime numbers greater than zero and less than or equal to some largest number `Max`.[3]

3.12 Write a program to test whether or not an input integer has the cube property. The cube property holds for a three digit number if the sum of the cube of the digits gives the number itself. A correct number is $153 = 1^3 + 5^3 + 3^3$.[3]

3.13 *GCD.* Write a program that finds the greatest common denominator *GCD* using a `while` loop. Write a *recursive* program that finds the *GCD.*

3.14 *LCM.* Write a program that finds the least common multiple *LCM* using a `while` loop. Write a *recursive* program that finds the *LCM.*

3.15 *Perfect Numbers.* A number P is said to be *perfect* if P is equal to the sum of all its factor *excluding itself.* Example, 28 is perfect.

3.16 Using the definition of perfect number, find all the perfect numbers in a range of numbers.

3.17 Two positive integers are *friendly* if each one is equal to the sum of the divisors (including one and excluding the number itself) of each other. 220 and 284 are friendly.

3.18 *Goldbach's Conjecture.* Goldbach's Conjecture is that any even number is the sum of two primes. Write a program to test Goldbach's Conjecture over a range of numbers.

3.19 Write a program to test the conjecture that the product of two consecutive numbers is a number between two odd numbers of which one numbers must be prime. Example: $7 \times 8 = 56$ and 56 is between 55 and 57. 57 is prime.

3.20 *Consecutive Primes.* Two odd numbers odd numbers that are also prime are called *consecutive primes.* Write a program to look for all consecutive primes in a range.

3.21 *Pythagorean Triples.* Three integers are *Pythagorean triples* if they satisfy the property that the square of the largest is equal to the sum of the squares of the other two numbers.

3.22 *Euclidean Remainder Algorithm.* The Euclidean division algorithm can be used to find the greatest common divisor of two numbers. Let a, b be two positive integers and $a < b$. Then $a = qb + r$ for some q and r. We create a sequence until the remainder is zero.

$$\begin{aligned} a &= q_0 b + r_0 \\ b &= q_1 r_0 + r_1 \\ r_0 &= q_1 r_1 + r_2 \\ r_1 &= q_1 r_2 + r_3 \\ &\cdots \end{aligned}$$

Write a program implementing this algorithm.

3.23 *Palindromes.* A *palindrome* is a number that reads the same forwards and backwards, like 12321. Write a program that tests input integers for the palindrome property. **Hint**.This should not be done with the numbers in their input (character) format.

3.24 *Sieve of Eratosthenes.* The sieve is used to find lists of prime numbers in a given interval that starts at 0. The first non-trivial prime is 2. Starting at two we proceed by removing all multiples of two from the list. The next prime, 3, is the lowest number not removed. Then all of 3's multiples are eliminated, and so on until we have found the largest prime in the interval. Write a sieve program.

4 Computer Arithmetic

4.1 Find the machine parameters for FLOATING POINT NUMBERs on your machine. This may be found in the technical documentation. Verify that these parameters are correct.

4.2 *Machine Epsilon.* Find the FLOATING POINT NUMBER `epsi` that has the the following properties:

(a) `1.0+epsi` is greater than `1.0` and

(b) Let `m` be any number less than `epsi`. Then `1.0+m` is equal to `1.0`.

`epsi` is called *machine epsilon*. It is of great importance in understanding floating point numbers.

4.3 Find the smallest and largest representable numbers for FLOATING POINT NUMBERs on your machine. This can be done from the definitions of the parameters. Verify that your numbers are correct.

4.4 *Casting Out 9s.* Every integer has the property that it is divisible by 9 if and only if the sum of its digits is dvisible by 9. Write a program to test an input integer for this property.

4.5 *Multiprecision Arithmetic.* Write a series of routines that will allow you to compute integers at twice the current capacity of the machine. **Hint**.For most current machines, the maximum positive integer is $2^{32}-1$. Therefore, the goal of this project is to be able to have a maximum of $2^{64}-1$

5 Simple Mathematical Functions

5.1 The geometric series is the series with the form $a, ar, ar^2, ar^3, \ldots$. Determine the *closed form solution* of the sum of the first n terms of the series. Write a program to computationally solve for the sum. Compare the closed form solution and the compute solution for $a = 1$ and $r = 1/16$ for n varying as $1, 2, \ldots, 20$. What do you observe?[3]

5.2 S_n is the partial sum defined by the series:

$$\begin{aligned} S_n &= \frac{1}{1^3} - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} \cdots \\ &= \Sigma_{k=0}^{n} (-1)^k \frac{1}{(2n+1)^3}. \end{aligned}$$

(a) Write a program to tabulate n and S_n by summing in the forward direction for $n = 1, \ldots, 50$.

(b) Write a program to tabulate n and S_n by summing in the backward

(c) What do you observe?

[3]

5.3 Produce a table of function values for the below:

(a) $x^3 + x^2 - 37x + 35$ in the interval $[-5, 5]$ in steps of 0.1.

(b) Repeat the above exercise with step size of $1/8$. Compare with above. What do you observe?

(c) e^{-x} for x in the interval $[0, 1]$ in steps of 0.001.

5.4 Compute $x^3 + x^2 - 37x + 35$ at $x = 0.5$ by two methods. The first method is to compute x^2 and x^3 then compute the polynomial. The second method is to use Horner's rule. Compare answers. What is the difference in operation count?

6 Limits, Sequences, and Series

6.1 Compute the geometric series from both the summation and the summation formula. Compare the two answers.

6.2 Write programs to study the following limits

(a) $\lim_{t\to 2} [\cos 2t]^{\frac{1}{t^2}}$

(b) $\lim_{t\to 2} \left[\frac{1+t}{t} - \frac{1}{\log(1+t)}\right]$

(c) $\lim_{t\to 2} \frac{\sin t}{t}$

6.3 Expand the following functions into their Maclaurin series and compute: sin x, cos x, e^x. **Hint**.Look up the series in the CRC handbook[2] or in Abramowitz and Stegun[1].

7 Analytic Geometry

7.1 Calculate the area of a triangle with sides a, b, and c using the following system:

$$\begin{aligned} s &= \text{semiperimeter} = (a+b+c)/2 \\ a &= \text{area} = \sqrt{s(s-a)(s-b)(s-c)}. \end{aligned}$$

[3]

7.2 Write a program to explore *polar coordinates* on the unit circle. Given x, y, the polar coordinates are given as

$$\begin{aligned} r &= \sqrt{(x^2+y^2)} \\ \theta &= \arctan y/x \end{aligned}$$

The inverse is given by

$$\begin{aligned} x &= r\cos\theta \\ y &= y\sin\theta \end{aligned}$$

Move around in the first quadrant for x between zero and one in steps of $1/32$. First find y, then convert the x, y-pair to polar and then back to rectangular. Compare the original x, y with the final x, y.

8 Polynomials

8.1 Write a set of routines to support polynomial arithmetic using arrays.

9 Zero Finding

9.1 *Newton-Raphson Square Root.* Let a be a number we want the square root of. Write a program using the following difference definition.

$$\begin{aligned} x_0 &= \frac{a}{2} \\ x_n &= \frac{1}{2}(x_{n-1} + \frac{a}{x_n}). \end{aligned}$$

[3]

9.2 *Quadratic Formula.* Consider the general quadratic polynomial ax^2+bx+c. Use the quadratic formula to find the two roots. Test your algorithm on two special situations: one wherein the roots are equal and one where one root is close to the smallest representable number and the other is close to the largest representable number.

9.3 For the following equations, do the following:

(a) Plot the function so as to show the zeroes.

(b) Using the graph estimate the root values to three digits.

(c) Show the above are correct estimates.

(d) Find the roots via a search program.

(e) Use any (preferably all) root finding methods to get the exact root:

i. bisection

ii. secant

iii. newton-raphson

iv. brent[1]

Compare results of the various methods.

$$\begin{aligned}
x^4 - x^3 - 3x^2_5x - 2 &= 0 \\
e^x - 3.5 &= 0 \\
\sin\theta - \theta = 0.5 & \\
x^4 + 4x^3 + 6x^2 + 4x + 1 &
\end{aligned}$$

10 Numerical Integration

11 Searching

12 Sorting

13 Statistics

14 Vectors, Matrices, and Numerical Linear Algebra

15 Deterministic Simulation

16 Stochastic Simulations

17 Graph Theory

18 Computational Science

19 Inverse Programming: What does it produce?

19.1 What does this produce[3]

⟨*Exercise 1*⟩≡

```
whatisit := 0
i := 1
while i <= 15 do
   whatisit := whatisit + i
   i := i +1
write whatisit
stop
```

[1]This is discussed in [4]. The software is available on `netlib`.

20 Advanced Applications

20.1 Program the *Game of Life*[8, 5, 6]

21 Glossary

21.1 <u>Floating Point Number</u>. Our view of floating point numbers is given in [7]. Briefly, the floating point numbers are zero and all the numbers of the form

$$s = m \times b^e = b^e \sigma_{i=1}^{t} a_i b^{-i}.$$

where b is the base, t is the number of *mantissa* units, and e is an integer in the range $[e_{min}, e_{max}]$. For most IEEE systems, single precision (`float` in `C`) $b = 2, t = 24, e_{min} = -128, e_{max} = 127$. There are three numbers of interest: the *machine epsilon*, and the largest, and the smallest representable numbers.

21.2 <u>Screen</u>. The <u>screen</u> is the output device for user direct viewing. In Unix, this would be `stdout`.

References

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables.* U. S. Government Printing Office, 1964. Updated by Yudell L. Luke in 1975.

[2] Chemical Rubber Company. *Standard Mathematical Tables.* Chemical Rubber Company, 13 edition, 1964.

[3] V. A. Dyck, J. D. Lawson, and J. A. Smith. *Introduction to Computing: Structured Problem Solving in WATFIV-S.* Reston Publishing Company, 1979.

[4] G. E. Forsythe, M. A. Malcum, and C. B. Moler. *Computer Methods for Mathematical Computation.* Prentice-Hall, 1977.

[5] Martin Gardner. Mathematical games. *Scientific American*, 224(2):112–117, February 1971.

[6] Martin Gardner. Mathematical games. *Scientific American*, 223(10):120–123, October 1971.

[7] D. E. Stevenson. Science, computational science, and computer science: At a crossroads. *Comm. ACM*, 37(12):85–96, 1994.

[8] Charles Wetherell. *Etudes for Programmers in C.* Prentice-Hall, 1978.