Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Theoretische Informatik
Dr. B. Gärtner, Prof. Dr. J. Hromkovič

27. November 2007

Informatik für Mathematiker und Physiker Serie 10 HS 07

URL: http://www.ti.inf.ethz.ch/ew/courses/Info1_07/

Aufgabe 1 [Schnellübung – 20 Min.]

a) (6 Punkte) Nehmen sie an, dass ein Programm, wenn es endet, nur die Rückgabe JA oder NEIN liefern kann. Beantworten Sie folgende Fragen:

Ist es möglich ein Programm zu schreiben, das...

(i) immer mit JA antwortet?

(ii) ein anderes Programm P auf der Eingabe x simuliert und dessen Rückgabewert zurück gibt, falls P auf der Eingabe x hält? Die Eingabe x sei von der Form, die von P erwartet wird.

(iii) für die Eingabe (x, i) angibt, ob x dem i-ten Programm P_i entspricht? Die Eingabe x entspricht einem Quellcode, also Text, und die Eingabe i ist eine positive ganze Zahl.

Ihre Antworten sollten eine kurze Begründung enthalten. Das heisst, falls die Antwort Ja lautet, beschreiben Sie wie Sie dieses Programm realisieren würden respektive welches ihnen bekannte Programm die entsprechende Aufgabe bewältigt. Falls die Antwort Nein lautet, geben Sie die Beweisidee an, weshalb es ein solches Programm nicht geben kann.

b) (2 Punkte) Beschreiben Sie kurz ein Problem, das nicht algorithmisch lösbar ist. Es darf jedoch kein Problem sein, das in a) schon genannt wird. Es ist kein Beweis erforderlich, dass das von Ihnen beschriebene Problem tatsächlich unlösbar ist.

Skript-Aufgabe 82 (4 Punkte)

Find pre- and postconditions for the following recursive functions.

a)
```
bool f (int n)
{
  if (n == 0) return false;
  return !f(n-1);
}
```

b)
```
void g (unsigned int n)
{
  if (n == 0) {
    std::cout << "*";
    return;
  }
  g(n-1);
  g(n-1);
}
```

c)
```
unsigned int h (unsigned int n, unsigned int b) {
  if (n == 1) return 0;
  return 1 + h (n / b, b);
}
```

Skript-Aufgabe 85 (4 Punkte)

In how many ways can you own CHF 1? Despite its somewhat philosophical appearance, the question is a mathematical one. Given some amount of money, in how many ways can you partition it using the available denominations (bank notes and coins)? The denominations in CHF are 1000, 200, 100, 50, 20, 10 (banknotes), 5, 2, 1, 0.50, 0.20, 0.10, 0.05 (coins). The amount of CHF 0.20, for example, can be owned in four ways (to get integers, let's switch to centimes): $(20), (10,10), (10,5,5), (5,5,5,5)$.

Solve the problem for any given input amount, by writing a program `partition` that defines the following function (all values to be understood as centimes).

```
// PRE: [first, last) is a valid nonempty range that describes
//      a sequence of denominations d_1 > d_2 > ... > d_n > 0
// POST: return value is the number of ways to partition amount
//       using denominations from d_1, ..., d_n
unsigned int partitions (unsigned int amount,
                         unsigned int* first,
                         unsigned int* last);
```

Use your program to determine in how many ways you can own CHF 1, and CHF 10. Can your program compute the number of ways for CHF 50?

Skript-Aufgabe 87 (4 Punkte)

Rewrite the following recursive function in iterative form and test with a program whether your iterative version is correct. What can you say about the runtimes of both variants for values of n up to 100, say?

```
unsigned int f (unsigned int n)
{
  if (n <= 2) return 1;
  return f(n-1) + 2 * f(n-3);
}
```

Skript-Aufgabe 88 (4 Punkte)

The following function finds an element with a given value x in a sorted sequence (if there is such an element).

```
// PRE: [first, last) is a valid range, and the elements *p,
//      p in [first, last) are in ascending order
// POST: return value is a pointer p in [first, last) such
//       that *p = x, or the pointer last, if so such pointer
//       exists
int* binary_search (int* first, int* last, int x)
{
  int n = last - first;
  if (n == 0) return last;          // empty range
  if (n == 1)
    if (*first == x)
      return first;
    else
      return last;
  // n >= 2
  int* middle = first + n/2;
  if (*middle > x) {
    // x can't be in [middle, last)
    int* p = binary_search (first, middle, x);
    if (p == middle)
      return last; // x not found
    else
      return p;
  } else
    // *middle <= x; we may skip [first, middle)
    return binary_search (middle, last, x);
}
```

What is the maximum number $T(n)$ of comparisons between sequence elements and x that this function performs if the number of sequence elements is n? Try to find an upper bound on $T(n)$ that is as good as possible. (You may use the statement of exercise 89 of the lecture notes.)

Challenge

You can solve the challenge exercise 92 from the lecture notes. It will be awarded a maxiumum of 8 points, and thus replaces two of the normal exercises.

Abgabe: Bis 4. Dezember 2007, 15.15 Uhr.