Midterm Examination 2
GEM 1501: Problem Solving for Computing

Wednesday 06.04.2011, duration half an hour

Matriculation Number: _______________

Rules
This test carries 12 marks and consists of 6 questions. Each questions carries 2 marks; full marks for a correct solution; a partial solution can give a partial credit.

Question 1 [2 marks].
Let $\varphi_0, \varphi_1, \ldots$ be the list of all syntactical correct Java Script functions with one input variable where the only data type used are natural numbers. Rice's Theorem says that every index sets is either $\emptyset$ or $\{0, 1, 2, \ldots\}$ or undecidable. Explain what an index set is and give an example of an undecidable index set.

Solution. An index set is a set I of indices such that for all indices d, e of the same partial function, either $d, e \in I$ or $d, e \notin I$. Here d, e are indices of the same partial function iff for all x, either both $\varphi_d(x)$ and $\varphi_e(x)$ do not halt or both halt with the same output. There are many index sets, here some examples:

- The index set $\{e : \exists x\, [\varphi_e(x) \text{ is defined}]\}$ of all programs which halt on some input;
- The index set $\{e : \forall x\, [\varphi_e(x) \text{ is defined}]\}$ of all programs which halt on all inputs;
- The index set of all programs which halt on at least 10 inputs;
- The index set of all programs e where $\varphi_e(0)$ and also $\varphi_e(\varphi_e(0))$ are both defined;
- The index set of all programs e of partial functions where $0, 1, 2, \ldots, 98, 99$ is in the range of φ_e.

All these index sets are undecidable by Rice's Theorem. Furthermore, the first, third, fourth and fifth are r.e. and the second is not. Most students gave the second as an example.

Question 2 [2 marks].
Consider the following finite automaton accepting ternary strings, that is, strings consisting of the digits 0, 1 and 2. The set of states of the finite automaton is $\{a, b, c, d, e\}$ and a is the starting state; b, c, d, e are the accepting states.

state	acc/rej	succ at 0	succ at 1	succ at 2
a	reject	a	b	c
b	accept	b	c	d
c	accept	c	d	e
d	accept	d	e	a
e	accept	e	a	b

Which of the following ternary strings is accepted by the automaton?
$\boxed{\text{x}}$ 211, $\quad$ ☐ 1121, $\quad$ ☐ 10202.

Please give a verbal description when a ternary string $x_0x_1x_2 \ldots x_n$ is accepted by the automaton. What properties must the digits $x_0, x_1, \ldots, x_n$ have?

Solution. If one orders the states of the automaton in a circle of the form $a - b - c - d - e - a$ then one sees, that the automaton on input 0 stays at its position, on input 1 advances by one position and on input 2 advances by two positions. Hence it returns to the starting position a whenever $x_0 + x_1 + \ldots + x_n$ are a multiple of 5 (as the circle has length 5). The automaton accepts the input whenever it ends up in b, c, d or e after processing the input; hence the automaton accepts all strings $x_0x_1x_2 \ldots x_n$ where the sum $x_0 + x_1 + x_2 + \ldots + x_n$ is not a multiple of 5, that is, is not an element of $\{0, 5, 10, 15, \ldots\}$. Accepted strings must have at least one digit. Examples of accepted strings are 211, 00100, 011, 0101, 10201 and examples of rejected strings are 1121, 10202, 00000, 22222, 1212121212 and 212.

Question 3 [2 marks].
A researcher proposes the following parallel sorting network for four inputs:

- Input (a_1, a_2, a_3, a_4);
- $b_1 = \min\{a_1, a_2\}$, $b_2 = \min\{a_3, a_4\}$, $b_3 = \max\{a_1, a_2\}$, $b_4 = \max\{a_3, a_4\}$;
- $c_1 = \min\{b_1, b_2\}$, $c_2 = \max\{b_1, b_2\}$, $c_3 = \min\{b_3, b_4\}$, $c_4 = \max\{b_3, b_4\}$;
- Output (c_1, c_2, c_3, c_4).

Is (c_1, c_2, c_3, c_4) always a sorted copy of (a_1, a_2, a_3, a_4)?
☐ Yes; ☒ No.

For answer "yes", explain why the network is correct; for answer "no", provide a counterexample which is not properly sorted.

Solution. One can see that $c_2 = \max\{\min\{a_1, a_2\}, \min\{a_3, a_4\}\}$. In the case that the two smallest elements are in a_1, a_2 at the beginning, c_2 will take the minimum of a_3 and a_4. So, for example, $(1, 2, 3, 4)$ is sorted to $(1, 3, 2, 4)$ and so the algorithm does even not preserve a correct order.

Additional Explanation. It could be noted that it has c_1 and c_4 correct. One can therefore add a further line with $d_1 = c_1$, $d_4 = c_4$ and $d_2 = \min\{c_2, c_3\}$ and $d_3 = \max\{c_2, c_3\}$ in order to obtain a sorted copy (d_1, d_2, d_3, d_4) of (a_1, a_2, a_3, a_4).

Question 4 [2 marks].
What are Monte Carlo algorithms and Las Vegas algorithms? What is the relation between these two types of algorithms: Is there a known method to adjust a Monte Carlo algorithm to a Las Vegas algorithm or is there a known method to adjust a Las Vegas algorithm to a Monte Carlo algorithm?

Solution. A Monte Carlo algorithm is an algorithm which is always fast and furthermore with high probability correct; a Las Vegas algorithm is an algorithm which is always correct but only with high probability (and not always) fast. One can convert a Las Vegas algorithm into a Monte Carlo algorithm, but no known method for the converse direction exists.

Additional Explanation. Note that Monte Carlo algorithms exist also for problems which are outside RP. So giving the definition of RP was not the requested answer. However, there is a connection. If a set A is in RP then it has a Monte Carlo algorithm deciding its membership in polynomial time. Furthermore, if A and its complement are both in RP then A has a Las Vegas algorithm deciding its membership.

Question 5 [2 marks].
Assume that a sequence is given by

$$f(0) = 0,\ f(1) = 1 \text{ and } f(n+2) = f(n+1) - f(n).$$

Furthermore, assume that adding, subtracting, multiplying, dividing and taking remainders have cost $O(1)$. What is the complexity to compute f in this model? Give the optimal value. Here the parameter n is just the number used as input for $f(n)$.

[x] $O(1)$ [ ] $O(\log(n))$ [ ] $O(n)$ [ ] $O(2^n)$.

Justify your choice: why can the given bound be obtained? A program with a runtime estimate is fine. In the case that you do not take $O(1)$, say why your bound is optimal and cannot be improved.

Hint: Calculate some values $f(0), f(1), f(2), \ldots$ of the sequence before answering this question.

Solution. The following is a table of the first 10 values of f.

n	0	1	2	3	4	5	6	7	8	9
f(n)	0	1	1	0	-1	-1	0	1	1	0

For $n = 0$ one can see that $f(n+6) = f(n), f(n+7) = f(n+1)$ and $f(n+8) = f(n+2)$. The task is now to show by induction that this holds for all n. Assuming it for n, one has $f((n+1)+6) = f(n+7) = f(n+1)$, $f((n+1)+7) = f(n+8) = f(n+2) = f((n+1)+1)$ and $f((n+1)+8) = f((n+1)+7) - f((n+1)+6) = f((n+1)+1) - f((n+1)) = f((n+1)+2)$. So the same statement holds for $n+1$ in place of n.

A consequence of the genral rule that $f(n+6) = f(n)$ is that $f(n) = f(n\%6)$ for all n. Therefore the following algorithm computes $f(n)$.

```
function f(n)
  { var m = n%6;
    switch(m)
      { case 0: return(0); break; case 1: return(1); break;
        case 2: return(1); break; case 3: return(0); break;
        case 4: return(-1); break; case 5: return(-1); break; } }
```

Taking the remainder is $O(1)$ according to the definition in the question. From then onwards, all depends on m which takes one of the values $0, 1, 2, 3, 4, 5$ and therefore the remaining part is also $O(1)$. It is just a table lookup.

Question 6 [2 marks].
An array x is a subsequence of an array y if one can obtain x by deleting some entries from y without changing the order of the remaining entries. For example, $(2, 8, 7, 9)$ is a subsequence of $(1, 2, 3, 8, 7, 6, 9)$ but $(2, 8, 7, 9)$ is not a subsequence of $(9, 8, 7, 6, 5, 4, 3, 2, 1)$. Furthermore, every x is a subsequence of itself. Write a program which returns 1 if the input array x is a subsequence of the input array y and which returns 0 otherwise; the entries in the arrays x and y are numbers.

Solution. The idea is to scan through the words x and y in parallel and to find positions $m_0, m_1, \ldots, m_{x.length-1}$ in y where $x[n]$ equals $y[m_n]$. Furthermore, $m_n < m_{n+1}$ for all $n < x.length - 1$. For the implimentation, it is not necessary to store the different values m_n premanently, one just makes an m going up in parallel to n and whenever $x[n]$ is detected to be equal to $y[m]$ in this loop then m has the value m_n as described above. Here is the program.

```
function issubsequence(x,y)
  { var r; var n = 0; var m = 0;
    while ((x.length>n) && (y.length>m))
      { if (x[n]==y[m]) { n++; }
        m++; }
    if (n == x.length) { r = 1; } else { r = 0; }
    return(r); }
```

Note that at the definition of a subsequence, the matching positions can be spread out as in the subsequence $(1, 3, 5)$ of $(0, 1, 2, 3, 4, 5, 6, 7, 8)$. This is the main difference between subsequences and substrings.

END OF PAPER