Midterm Examination 2
GEM 1501: Problem Solving for Computing

04.04.2007, 12.00-12.30h

Matriculation Number: ____________________

Rules
Each correct question, 1 mark. Maximum score: 12 marks.
Programming Language for Questions 7–11 is Java Script.

Question 1. Solve the following problem using resolution. Do each step as prescribed. The logical variables are x_1, x_2, x_3, x_4, x_5, write the clauses after each step.

Given clauses: $\neg x_1 \vee x_2$, $\neg x_1 \vee x_3$, $x_1 \vee x_3$, $\neg x_4 \vee \neg x_5$, $x_4 \vee x_5$.

Resolve x_1, new clauses:
$\underline{\;x_2 \vee x_3\;}$, $\underline{\;x_3\;}$, $\underline{\;\neg x_4 \vee \neg x_5\;}$, $\underline{\;x_4 \vee x_5\;}$.

Choose x_2 as ☐ false ☒ true, new clauses:
$\underline{\;x_3\;}$, $\underline{\;\neg x_4 \vee \neg x_5\;}$, $\underline{\;x_4 \vee x_5\;}$.

Choose x_3 as ☐ false ☒ true, new clauses:
$\underline{\;\neg x_4 \vee \neg x_5\;}$, $\underline{\;x_4 \vee x_5\;}$.

Resolve x_4, new clause: $\underline{\;\neg x_5 \vee x_5\;}$.

This instance of the 2SAT problem is ☒ satisfiable ☐ not satisfiable.

Question 2. Which three of the following problems are known to be NP-complete; note that NP-complete problems are in NP:
☐ 2SAT, ☒ 3SAT, ☒ 4SAT, ☒ Monkey Puzzles,
☐ $n \times n$ Checkers Winning Strategy, ☐ Presburger Arithmetic.

Question 3. The following problems are all semantic problems and undecidable by the Theorem of Rice. Which one of them is still recursively enumerable?
☐ The set of all texts for programs which halt on all inputs;
☒ The set of all texts for programs which halt on at least two inputs;
☐ The set of all texts for programs which halt on infinitely many inputs.

Question 4. Which of the following commands can be added to Counter Programs such that all functions computable in polynomial time can be computed by this enhanced counter program in polynomial time, but nothing more. Every basic operation of a counter program counts as one step, including the added operations.

- [x] Compare variables and jump ("if $x < y$ then goto ...");
- [x] Add variables ("let $x = y + z$");
- [x] Subtract variables ("let $x = y - z$");
- [ ] Multiply variables ("let $x = y * z$");
- [ ] Compute powers of variables ("let $x = y^z$").

Mark three out of five boxes.

Question 5. What is the product complexity of a parallel sorting algorithms? Please mark the appropriate box.

- [ ] The product of the parallel time and sequential space of sorting;
- [x] The product of the parallel time and the number of processors used;
- [ ] The product of the parallel time and the sequential time of the algorithm.

What is the product complexity of the optimal sorting network? Mark the best possible answer (mark exactly one box):

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

Question 6. Give the definition of the set of compressible texts.

The set of all compressible texts contains a text T iff there is a Java Script program P such that P is shorter than T, P outputs T and P does not request any input.

What is the special property of this set which makes it theoretically interesting? Mark the appropriate box.

- [ ] It is not recursively enumerable but its complement is;
- [ ] It is recursively enumerable and complete;
- [x] It is recursively enumerable and incomplete and undecidable;
- [ ] It is decidable but not in solvable in exponential time.

Question 7. Look at the following sample JavaScript program.

```
function fact(n)
  { var m = Math.round(n);
    if (m>1) { return(m*fact(m-1)); }
    return(1); }


var n;
do { n = window.prompt("Input a number",22-18);
     window.alert("The factorial of "+n+" is "+(fact(n))+"."); }
   while(window.confirm("Do you want to go on?"));
```

What is the output of the program of the second window if the user just presses return in the first window?

The factorial of 4 is 24.

How does the user terminate the loop in the third window?

The user clicks "cancel" on third window.

Question 8. Optimize the following loop such that only 4 times a variable is mentioned in each run of the loop (instead of 8 times now)! Write the complete new function.

```
function loopy(u)
  { var v = 0; var w = 0;
    while (v<u) { w = w+v*v; v=v+1; }
    return(w); }
```

Two possible solutions:

```
function loopy(u)
  { var v = u; var w = 0;
    while (v--) { w += v*v; }
    return(w); }

function loopy(u)
  { var v = u; var w = 0;
    while (v>0) { v--; w += Math.pow(v,2);  }
    return(w); }
```

Question 9. Analyze the following Java Script function.

```
function stringtest(x)
  { var counta = 0; var countb = 0;
    var y = x.length;
    while (y>0)
      { y=y-1;
        if (x.charAt(y) == "a") { counta++; }
        if (x.charAt(y) == "b") { countb++; } }
    if (counta < countb)
      { return("Accept"); }
    else
      { return("Reject"); } }
```

This function accepts words where there are

☐ more "a" than "b" ☐ as many "a" and "b" ☒ more "b" than "a".

Furthermore, the existence of other characters (like "c", "d", ...)

☒ has no influence ☐ causes "Accept" ☐ causes "Reject".

The function can also be implemented by

☐ a finite automaton ☒ an one-stack automaton ☒ a Turing machine.

The last question can have several or no boxes marked, for the first two, mark exactly one box.

Question 10. Write in Java Script any function which cannot be computed by a one-stack automaton. No proof needed. Function must be syntactically correct.

Solution:

```
function f(x)
  { var n = x.length; var m = "Accept"; // value of m can be modified
    var counta = 0; var countb = 0; var countc = 0;
    while (n>0)
      { n--;
        if (x.charAt(n) == "a") { counta++; }
        if (x.charAt(n) == "b") { countb++; }
        if (x.charAt(n) == "c") { countc++; } }
    if ((counta != countb) || (counta != countc) || (countb != countc))
      { m = "Reject"; }
    return(m); }
```

Question 11. Which of the following Java Script functions can be parallelized efficiently?

```
function ga(x)
  { var n = x.length; var y = 0; var m;
    for (m=0;m<n;m=m+1)
      { y = y+x[m]; }
    return(y); }

function gb(x)
  { var n = x.length; var y = 0; var m;
    for (m=0;m<n;m=x[m])
      { y = y+m; }
    return(y); }

function gc(x)
  { var n = x.length; var y = 0; var m;
    for (m=0;m<n;m=m+1)
      { y=y+1; x[m] = y; }
    return(y); }

function gd(x)
  { var n = x.length; var y = 0; var m;
    for (m=0;m<n;m=x[m])
      { y=y+1; x[m] = y; }
    return(y); }
```

Mark the two boxes where the function is parallelizable:

[x] ga, [ ] gb, [x] gc, [ ] gd.

Question 12. A sorting network consists of n processors each carrying a word. In even steps, processor numbers $2m$ and $2m+1$ exchange their words if they are in wrong order. In odd steps, processors number $2m+1$ and $2m+2$ exchange their words if they are in wrong order. Comparing this algorithm with known algorithms (like Bubble sort), what can be said about the complexity of the sorting network? Mark the best possible answer (mark exactly one box):

The number of rounds needed, that is, the parallel time, is

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