Principles of programming languages

Spring 2006

Sample Examination Paper (solutions)

This is a closed book written exam.
The answers can be given in Dutch or English.
The final grade is calculated as (Q1+Q2+Q3+…..+10)/10
A pass for this exam is valid only in combination with a fulfilled assignment

---

Credits:

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	SQi
a)	**3**	**4**	**5**	**10**	**10**	**5**	**5**	**5**	**5**	
b)	**2**	**4**	**5**			**5**	**3**	**6**	**5**	
c)	**2**						**3**			
d)	**3**									
Totaal	**10**	**8**	**10**	**10**	**10**	**10**	**11**	**11**	**10**	**90**

---

1. Syntax and semantics [10p]

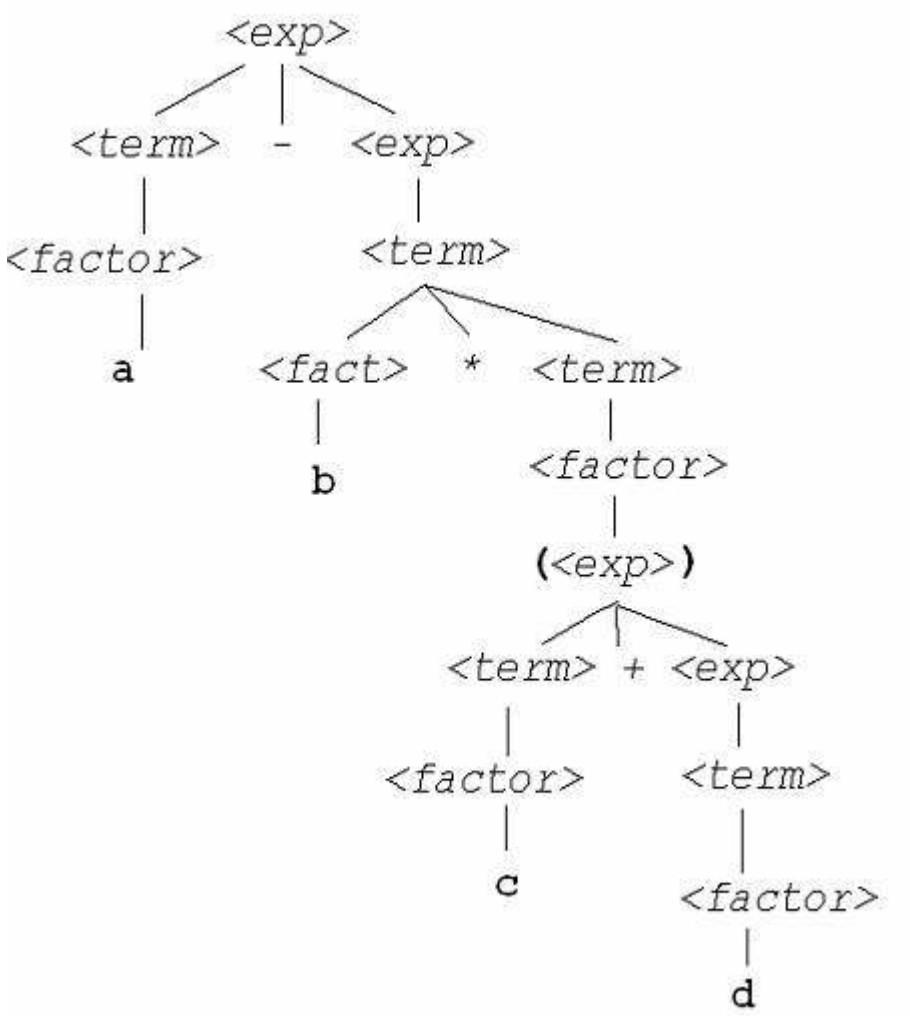
Given the following BNF grammar:
<*exp*> ::= <*term*> + <*exp*> | <*term*> – <*exp*> | <*term*>
<*term*> ::= <*factor*> * <*term*> | <*factor*> / <*term*> | <*factor*>
<*factor*> ::= (<*exp*>) | a | b | c | d | 1 | 2 | 3 | 4

a) Build a parse tree for the expression a-b*(c+d). **[3p]**
b) Show on the tree what a production is, a non-terminal symbol, a token. **[2p]**
c) What is the value of 1-2*3 in the language defined by the grammar, if expressions are evaluated in the order implied by the parse tree? **[2p]**
d) Which of the following expressions can NOT be parsed by the grammar? **[3]**

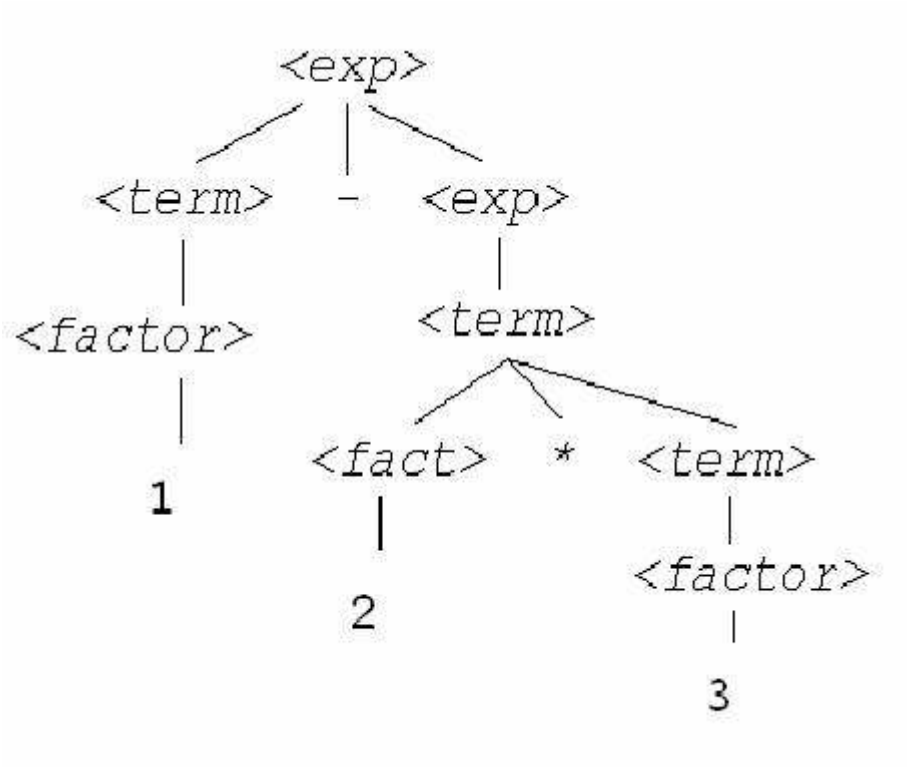
((a))
a+b+(c+d)
a+b*(-1)
a+b*c

---

a) See the parse tree. Non-terminal symbols are language constructs like <term> and <factor>, a production rule is used each time a new level in the tree is built, for example <exp>::=<term>-<exp>. Tokens are the smallest units of syntax, for example a and b.


<exp>
<term>
-
<exp>
<factor>
<term>
a
<fact>
*
<term>
b
<factor>
(<exp>)
<term>
+
<exp>
<factor>
<term>
c
<factor>
d

c) From the parse tree built for the expression 1-2*3 it can be seen that 2*3 is evaluated first and this intermediate result (6) is substracted from 1, finally resulting -5.



d) The expression containing -1.

---

2. Language systems [8 p]

a) List the steps in the classical sequence of a language system**.[4 p]**

b) What is the binding time for the meaning of the keyword while in Java? **[4p]**

---

a) source code file creation, compiling, assembling, linking, loading, running
b) language definition time

---

3. Types [10p]

a) Which of the following is an example of coercion **[5p]?**

A. The way the expression a+b is evaluated in C, when a is an int and b is a float.
B. The way the length function in ML works on any type of the form 'a list .
C. The way the + operator in C applies to pairs of integers and also to pairs of real numbers

b) What is a type annotation? Give an example in ML. **[5p]**

---

a) answer A.
b) with a type annotation the programmer supplies explicit type information to the language system.
ML example: fun prod (a,b:real) = a* b ;

---

4. Functional programming [10 p]

Explain what pattern matching is in ML and show how this applies for the following code snippet]:

```
    fun f nil = 0
| f (first::rest) = first + f rest


    f [1,3,5,7]
```

The definition of f uses a pattern matching style and contains 2 patterns: nil for the base case and the compound pattern first::rest for the recursive case.
In the function call f[1,3,5,7]the function f is applied to a list [1,3,5,7]. ML tries to match this list to one of the patterns used in the function definition. The list matches with first::rest. By pattern matching, ML can extract the first element of the list in the variable first and the rest of the list in variable rest. So now first = 1 and rest is [3,5,7] and the result is 1 + the result of applying function to [3,5,7] . The function is called recursively for [3,5,7], and pattern matching is applied again giving the result 1+ 3 + f [5,7]. And so on, until the rest is the empty list []. In this case ML matches it with the base case nil and the result is 0. Coming back from recursion, the final result is 1+3+5+7+0. So the function f returns the sum of all elements of a list.

---

5. Memory management [10p]

Comment on the trade-off between static and dynamic activation record allocation.

An activation record (AR) is a block of memory containing all activation specific data that a function needs. The AR can be allocated statically or dynamically.
Static allocation is a simple technique: allocate one allocation record for each function, always at the same address. The advantage is efficiency because allocation happens before the program runs. Disadvantage is that recursive calls and dynamic data structures are not allowed. Dynamic allocation happens at run time, the AR form a stack of frames in memory. AR are pushed on call and popped on return.
Dynamic data structures are allocate d in a special memory block called heap. Dynamic allocation allows recursion and dynamic data structures but is slower than static allocation and creates memory fragmentation problems.

6. Object oriented programming [10p]

Consider this C++ program :

```
#include <iostream>
using namespace std;

class A {
    public:
        void print() {cout << "A";}
};

class B : public A{
    public:
        void print() {cout << "B";}
};

int main()
{
    A* p ;
    p = new A() ;
    p->print() ;

    p = new B() ;
    p->print() ;
  return 0;
}
```

a) By default C++ provides static method binding. What is the output of this piece of code? **[5p]**
b) What would be the output if C++ used dynamic method binding instead? **[5p]**

a) class B inherits from class A. When using static method binding, everywhere will be used the print () method of the superclass A, so the result will be AA.
b) If dynamic binding is used, each object has its own print () method which is used. The result will be AB.

7. Exceptions [11p]

a) Name and explain the components of a Java exception handling mechanism **[5p]**
b) What is the output of this Java code? **[3 p]**

```
1   System.out.print("1");
2   try {
3       System.out.print("2");
4       if (true) throw new Exception();
5       System.out.print("3");
6   }
7   catch (Exception e) {
8   System.out.print("4");
9   }
10   finally {
11   System.out.print("5");
12   }
13   System.out.println("6");
```

c) What happens if we change in line 4 **new (Exception e)** with **new (Throwable e)**? **[3 p]**

---

a) Java exception mechanism contains a try statement which wraps the statement to be executed. If an error occurs when the statement is executed, Java creates an object that inherits Throwable class and throws this exception object. If the program does nothing to catch this exception, the program terminates and Java generates an error message. The exception can be caught by an exception handler, a catch statement. Finally statement is optional, but always executed.

b) 1 2 4 5 6
c) 1 2 5

Q8. Logic programming (Prolog) [11 p]

Given the facts:

```
loves(john, jane).
loves(bill, jane).
loves(james,jane).
loves(mary, bill).
loves(jane, james).
```

And the rule:

```
goodmatch(X,Y) :- loves(X,Y), loves(Y,X).
```

a) What is unification? Show how unification works in answering the question **[5p]**

```
?- loves (mary, X)
```

b) Show the backtracking involved in answering the question: **[6p]**

?- goodmatch(A,B)

---

a) 2 terms can be unified if a binding can be found that makes these terms equal. For example for the query asking who loves mary, loves (mary,Y), Prolog will try to unify the term loves (mary, X) with some given fact. Prolog succeeds to unify term (mary,X) with the given fact loves(mary, bill) and will return the binding which makes the two terms equal, X=bill.

---

b). The question goodmatch(A,B)is unified with the head of the rule and thus requires the subgoals: loves(A,B) and loves (B,A) to be satisfied.
The substitutions A ¬john and B¬jane allow the first subgoal to be matched with the first fact, but the second goals fails. The system backtracks to the first subgoal and tries A¬bill and B¬jane so that the second goal is matched. The second goal fails again. The system again backtracks and tries A¬james and B¬jane with satisfies both goals. System stops with the answer A=james, B=jane.

Q9. Scripting languages (Python) [10p]

a) Why is Python considered a very high level programming language? Support your answer with examples **[5p].**

b) Explain what this Python function does and give an example in which this function is used properly **[5p].**

```
def  func (*args):
    res = args[0]
    for arg in args[1:]:
        if arg < res:
          res = arg
    return res
```

---

a) Python has very powerful built-in data structures and methods which spare the programmer a lot of time, like lists, strings, dictionaries and methods like slicing, sorting, appending, deleting, replacing, mapping, etc. In this way the programmer does not spend time on low level implementation details by coding algorithms.

Give an example you liked.

```
>>> S=”xxxSPAMxxxSPAMxxx’
>>> S.replace (‘SPAM’,’EGGS’)
‘xxxEGGSxxxEGGSxxx’
```

b) This function accepts an arbitrary number of parameters and puts them in a tuple. The function returns in res the smallest element of the list of arguments given as input. A possible call could be func 10,15, 1, 4, 67. The result will be 1. But also func ‘a’, ‘aab’, ‘ac’ is possible