Seminar 5

You are expected to present solutions to these problems at the seminar in week 6.

1 The following program computes the greatest common divisor of n and m using recursion in a programming language that has methods but no parameters and return values.

```
1 void main() {
2   int n=15;
3   int m=9;
4   int result;
5   gcd();
6   void gcd() {
7     int diff=n-m;
8     if (diff==0) {
9       result=n;
10    } elseif (diff>0) {
11      n=diff;
12      gcd();
13    } else {
14      m=-diff;
15      gcd();
16    }
17  }
18 }
```

Assume that the computations can be made without temporary variables and that the execution model from lecture 10 is used. What is the contents of the stack of activation frames when it has maximal size, just before it is going to shrink? Assume that the stack starts at address 0 and use line numbers from the program for the contents of retaddr. Also specify the contents of FP and SP.

2 The following program decides in a primitive way if a natural number is even or odd.

```
void main() {
  int n=15;
  print(even(n));
  boolean even(int n) {
    if (n==0) return true;
    if (n==1) return false;
    return even(n-2);
  }
}
```

The procedure even is tail recursive, i.e. the recursive call appears at the very end of the procedure and the compiler may replace the recursion with iteration. What is the maximal number of procedure frames that will be on the stack if the compiler doesn't make the optimization? How many will there be if recursion has been eliminated?

3 A static method in Java can just use the static attributes of the class (and no instance attributes). A call of such a method can be implemented more efficiently than a nonstatic method. Explain why.

4 One possible list implementation in Java is given by:

```
interface List { }

class Empty implements List { }
```

```
class Node implements List {
  Object object;
  List next;
  Node(Object object, List next) {
    this.object=object;
    this.next=next;
  }
}

class Main {
  public static void main(String [] arg) {
    List list = new Node("node1", new Node("node2'', new Empty()));
  }
}
```

Describe, in principle, how list is represented on the stack and the heap.

5 The following program is written in a language where method declarations may be nested. Translate the method updateAccount to intermediate code as defined in lecture F11.

```
void program() {
  int updateAccount(int initialBalance) {
  int balance = initialBalance;
  int amount = read();
  while (amount <> 0) {
    balance = balance + amount;
    amount = read();
  }
  return balance;
}

int read() {
...
}

...
}
```

6 The following program is written in a language where method declarations may be nested. Translate the method compute to intermediate code as defined in lecture F10.

```
void program() {
  class PolynomDegree3 {
    int k0, k1, k2, k3;
    int compute(int x) {
      return (k0 + k1*x + k2*power(x,2) + k3*power(x,3);
    }
  ...
  }

  int power(int x, int y) {
    ...
  }
}
```