Prof. Dr. Peter Thiemann
Sergio Feo-Arenis
Sergiy Bogomolov

Summer Term 2014

Software Engineering

http://proglang.informatik.uni-freiburg.de/teaching/swt/2014/

Exercise Sheet 10

Exercise 1: Tracking Dependencies (10 Points)

Consider the following program. The function `read()` reads a number from the console and returns it. The function `write()` writes a number to the console.

```
void main()
{
    int a, b, sum, product;
    sum = 0;
    product = 1;
    a = read();
    b = read();
    while (a <= b)
    {
        sum += a;
        product *= a;
        a++;
    }
    write(sum);
    write(product);
}
```

Exercise 1.1: Effects of statements

Name for each statement in the above program the set of variables which are *read* and the set of variables which are *written* by the statement.

Exercise 1.2: Control-Flow-Graph

In a Control Flow Graph, nodes represent program locations and are labelled with statements. Edges are used to represent jumps. There is an edge from statement A to B iff there is an execution of the program where B executes directly after A.

Draw the Control Flow Graph of the above program. Use a dedicated entry node labelled 'Entry: main' and a dedicated exit node labelled 'Exit'.

Exercise 1.3: Control Dependencies

Based on the Control-Flow-Graph from Exercise 1.2 compute the control dependencies and visualize them with a graph. It should have the same set of nodes as the CFG whereas the edges are defined as follows: There is an edge from node A to node B iff B is control-dependent on A.

Exercise 1.4: Data Dependencies

Based on the Control-Flow-Graph from Exercise 1.2 compute the data dependencies and visualize them with a graph. It should have the same set of nodes as the CFG whereas the edges are defined as follows: There is an edge from node A to node B iff B is data-dependent on A.

Exercise 2: Fixing Defects (10 Points)

Look at the program below which takes a string and counts a number of occurancies of every English letter in it. The program throws an exception on input “the quick brown fox jumped over the lazy dog’s tail”. Apply dd_{min} to find a minimal input for which the program also throws an exception. Then apply the algorithm to locate defects and fix the defect.

```java
import java.util.Scanner;

public class stringAnalysis {

  public static void main(String[] args) {
    String s = new String();
    Scanner input = new Scanner(System.in);
    int i;

    System.out.println(``Enter a string:'');
    s = input.nextLine();

    int[] cnt = new int[25];

    s = s.toLowerCase();

    for (i = 0; i < s.length(); i++)
      if (Character.isLetter(s.charAt(i)))
        cnt[s.charAt(i) - 'a']++;

    for (i = 0; i < cnt.length; i++)
      if (cnt[i] != 0)
        System.out.println(``cnt [``+(char)('a'+i)+``]=``+cnt[i]);
  }
}
```