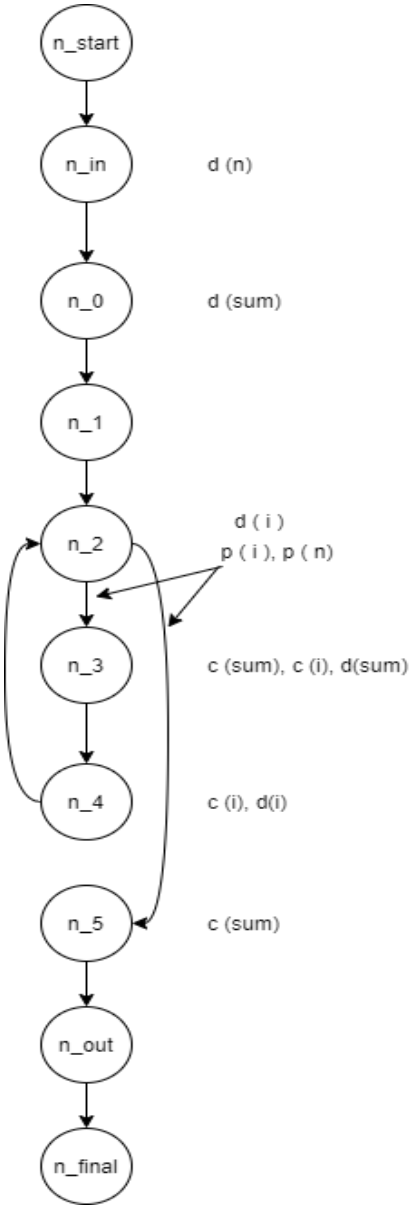
Software Quality Assurance (WS20/21)
Problem Set 3

Problem 1: Data Flow Oriented Test
Given is the function sum:

```
01  public static int sum(int n) {
02  int sum = 0;
03  int i;
04  for (i = 1; i <= n; i++) {
05    sum = sum + i;
06  }
07  return sum;
08 }
```



a) Please create a control flow diagram with data flow annotation for the function sum.
-

b) Write down all def-use pairs in a table as in the example below. Indicate p-uses and c-uses.
-

Use	Defined in	Path	Variable
p1	n_{in}	in, 0, 1, 2, 3	n
p2	n_{in}	in, 0, 1, 2, 5	n
p3	n_2	2, 3	i
p4	n_2	2, 5	i
p5	n_4	4, 2, 3	i
p6	n_4	4, 2, 5	i
c1	n_0	0, 1, 2, 3	sum
c2	n_0	0, 1, 2, 5	sum
c3	n_2	2, 3	i
c4	n_2	2, 3, 4	i
c5	n_3	3, 4, 2, 3	sum
c6	n_3	3, 4, 2, 5	sum
c7	n_4	4, 2, 3	i
c8	n_4	4, 2, 3, 4	i

c) Please determine the minimal necessary test path for fulfilling the **all defs criterion** of the sum function. Please denote the required test path and mark this path in the control flow diagram.
-

Minimal path: $n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_5, n_{out}$

d) Please determine the minimal necessary test path for fulfilling the **all c-uses criterion** for the sum function. Please denote the required test path and mark this path in the control flow diagram.
-

Minimal path: $n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_3, n_4, n_2, n_5, n_{out}$

e) Please determine the minimal necessary test path for fulfilling the **all p-uses criterion** for the sum function. Please denote the required test path and mark this path in the control flow diagram.

-

Minimal path: $n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_5, n_{out}$

f) Please determine the minimal necessary test path for fulfilling the **all c-uses/some p-uses criterion** for the sum function. Please denote the required test path and mark this path in the control flow diagram.

-

Minimal path: $n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_3, n_4, n_2, n_5, n_{out}$

Problem 2: Path Coverage Test

a) Please determine **the minimal necessary test cases** for fulfilling the structured path coverage test **for the parameter k=1** for the sum operation.

-

The structured path test distinguishes only path that execute a loop not more than k times. This avoids the explosion of the number of paths caused by loops.

K = 1 : $n_{start}, n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_5, n_{out}, n_{final}$ // n=1

b) Please determine the **minimal necessary test cases** for fulfilling the boundary interior test **for the sum operation**.

-

K = 0 : $n_{start}, n_{in}, n_0, n_1, n_2, n_5, n_{out}, n_{final}$ // n=0

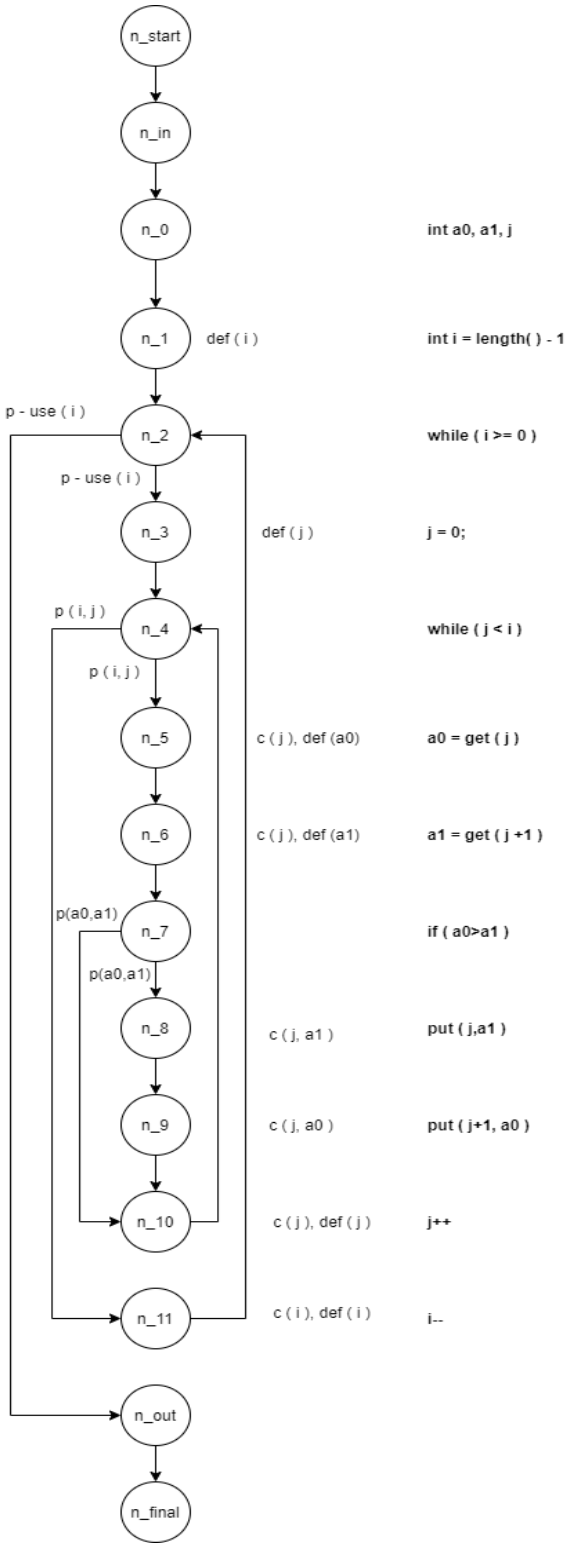
K = 1 : $n_{start}, n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_5, n_{out}, n_{final}$ // n=1

K = 2 : $n_{start}, n_{in}, n_0, n_1, n_2, n_3, n_4, n_2, n_3, n_4, n_2, n_5, n_{out}, n_{final}$ // n=2

Problem 3: Data flow oriented test

Given is a section of code for sorting a one-dimensional integer field using the bubble sort algorithm. The corresponding control flow diagram is presented as well.

```
01 public static int[] elements = {42,4,8,23,15,16};
02 public static int length() {return elements.length;}
03 public static int get(int i) {return elements[i];}
04 public static void put(int i,int x) {elements[i]=x;}
05 public static void bubblesort() {
06   int a0,a1,j;
07   int i=length()-1;
08   while (i>=0) {
09     j=0;
10     while (j<i) {
11       a0=get(j);
12       a1=get(j+1);
13       if (a0>a1) {
14         put(j,a1);
15         put(j+1,a0);
16       }
17       j++;
18     }
19     i--;
20   }
21 }
```



a) Draw the control flow diagram of the function bubblesort(), with the missing data flow attributes for a data flow oriented test. Give the minimal necessary test path for the fulfillment of statement coverage. Briefly explain the definition of the statement coverage criterion.

-

b) Give the **minimal necessary test path** for the fulfillment of **statement coverage.** Briefly explain the definition of the statement coverage criterion.

-

Coverage of all nodes, or at least single execution of all statements:

n_{start}, n_{in}, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_4, n_{11}, n_2, n_{out}, n_{final}

c) Give the minimal **necessary test path for the fulfillment of branch coverage.** Briefly explain the definition of the branch coverage criteria.

-

Coverage of all branches, or at least single execution of all true and false branches:

n_{start}, n_{in}, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_4, n_5, n_6, n_7, n_{10}, n_4, n_{11}, n_2 n_{out}, n_{final}

d) Fill out the following table **with all def-p-use** pairs P

P	Def	Path	Variable
p1	n1	n1, n2, n3	i
p2	n1	n1, n2, n_out	i
p3	n1	n1, n2, n3, n4, n5	i
p4	n1	n1, n2, n3, n4, n11	i
p5	n11	n11, n2, n3	i
p6	n11	n11, n2, n_out	i
p7	n11	n11, n2, n3, n4, n5	i
p8	n11	n11, n2, n3, n4, n11	i
p9	n3	n3, n4, n5	j
p10	n3	n3, n4, n11	j
p11	n10	n10, n4, n5	j
p12	n10	n10, n4, n11	j
p13	n5	n5, n6, n7, n8	a0
p14	n5	n5, n6, n7, n10	a0
p15	n6	n6, n7, n8	a1
p16	n6	n6, n7, n10	a1
p17			
p18			

e) Give the minimal necessary **def-use-pairs for the fulfillment of the all-p-uses/some-c-uses test**. Utilize the set of def-p-use-pairs identifiers in the table, e.g. {p1, p2}. Briefly state the definition of this criterion.

-

All p-uses/some-c-uses

If no p-use exists for a variable, add one c-use for it.

All P = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16} suffices, no c-uses needed.

Problem 4: State-based Test

Given is the specification of a digital watch software.
For adjustment of a digital watch, the following states are to be considered:

- Normal time: State after inserting the battery
- Adjust Hours: Hours can be adjusted
- Adjust Minutes: Minutes can be adjusted
- Adjust Seconds: Seconds can be adjusted

The following events could occur:

- Start signal: Battery inserted
- Button 1 pressed
- Button 2 pressed
- The two buttons must not be pressed simultaneously.

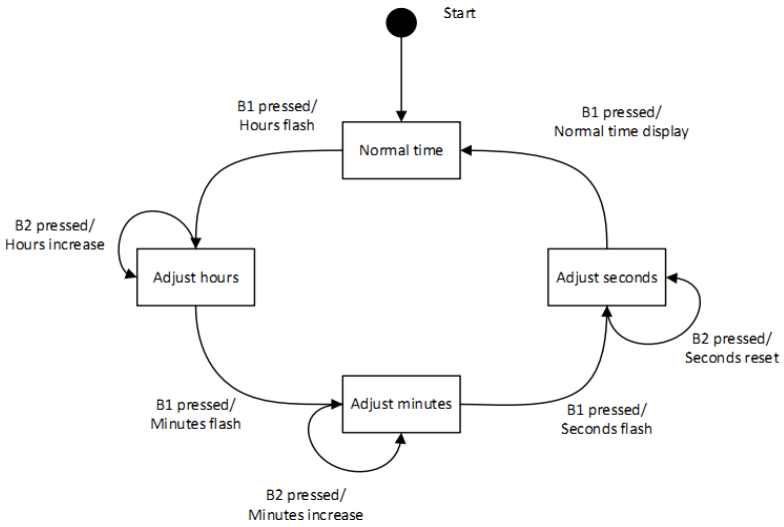


Figure 1: State chart "Watch adjustment "

The following outputs could happen:

- Hours flash: The operator is currently in the hour editing mode.
- Minutes flash: The operator is currently in the minute editing mode.
- Seconds flash: The operator is currently in the second editing mode.
- Hours increase: The hour display has increased by 1 hour.
- Minutes increase: The minutes display increases by 1 minute.
- Seconds reset: 00 displays as second display.
- Initialization: Display of 00:00:00

a) Please determine the **test data for the program execution that traverses every <u>state</u>**. Please select the simplest test cases.

-

normal time, button1 pressed > adjust hours, button1 pressed > adjust minutes, button1 pressed > adjust seconds

b) Please determine the test data for the program execution that **traverses every <u>transition</u>**. Please select the simplest test cases.

-

normal time, button1 pressed > adjust hours, button2 pressed > adjust hours, button1 pressed > adjust minutes, button2 pressed > adjust minutes, button1 pressed > adjust seconds, button2 pressed > adjust seconds, button1 pressed > normal time