**MT Exam CMSE-520 18.11.2024 (90 min, 25 points)**

St. Name, Surname\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ St.Id#\_\_\_\_\_\_\_\_\_\_\_\_\_

**Mobiles and calculators are not allowed. Three cheat sheets with your own handwritings can be used**

Instructor Alexander Chefranov

**4 questions, 6 pages**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **1** | **2** | **3** | **4** | **Total** |
| **Points** | **10** | **5** | **6** | **4** | **25** |
| **Grade** |  |  |  |  |  |

**Task 1. (10 points)** Consider the code below:

MODULEl-Implementation #l-a single procedure implements a “selection sort.”

MODULE1

package sort1 is

type array-type is array (l..lOOO) of integer;

-- Specifications of the procedure

procedure sort1 (n : in integer; to-be-sorted: in out array-type; a--or-d: in character);

end sortl;

-- The code in the body is evaluated

package body sort1 is

procedure sort1 (n : in integer; to-be-sorted: in out array-type; a-or-d: in character) is

location, temp : integer;

begin

for start in l..n loop

location := start;

--loop to get min or max each time

for i in (start + 1X.n loop

if a-or-d = ‘d’ then

if to-be-sorted(i) > to-be-sorted(location) then

location := i;

end if;

else

if to-be-sorted(i) < to-be-sorted(location) then

location := i;

end if;

end loop;

--The exchange

temp := to-be-sorted(start);

to-be-sorted(start) := to-be-sorted(location);

to-be-sorted(location) := temp;

end loop;

end sortl;

end sortl;

For the procedure sort1 above

1. **(3 points)** Build control flow graph
2. **(4 points)** Calculate cyclomatic complexity
3. **(3 points)** Specify respective independent paths

Hints:

|  |  |
| --- | --- |
| It is proposed to use a graph in which each exit point is connected back to the entry point. In this case, the graph is [strongly connected](https://en.wikipedia.org/wiki/Strongly_connected). Here, the cyclomatic complexity of the program is equal to the [cyclomatic number](https://en.wikipedia.org/wiki/Cyclomatic_number" \o "Cyclomatic number) of its graph (also known as the [first Betti number](https://en.wikipedia.org/wiki/Betti_number#Example_2:_the_first_Betti_number_in_graph_theory)), which is defined as[[2]](https://en.wikipedia.org/wiki/Cyclomatic_complexity#cite_note-mccabe76-2)  M=E−N+P.  where *E* = the number of edges of the graph, *N* = the number of nodes of the graph, *P* = the number of [connected components](https://en.wikipedia.org/wiki/Component_(graph_theory)).  This may be seen as calculating the number of [linearly independent cycles](https://en.wikipedia.org/wiki/Linearly_independent_cycle) that exist in the graph: those cycles that do not contain other cycles within themselves. Because each exit point loops back to the entry point, there is at least one such cycle for each exit point. | undefined  Each exit point is connected back to the entry point. This graph has 10 edges, eight nodes and one [connected component](https://en.wikipedia.org/wiki/Connected_component_(graph_theory)), which also results in a cyclomatic complexity of 3 (10 − 8 + 1 = 3). |
| **if** (c1())  f1();  **else**  f2();  **if** (c2())  f3();  **else**  f4();  The cyclomatic complexity of the program is 3 (as the strongly connected graph for the program contains 9 edges, 7 nodes, and 1 connected component) (9 − 7 + 1). | undefined  S  A  F  E  D  C  B |

There are 4 cycles (paths) but only 3 of them are linearly independent. McCabe developed an algorithmic procedure (called the “baseline method”) to determine a set of basis paths. The method begins with the selection of a “baseline” path, which should correspond to some “normal case” program execution. This can be somewhat arbitrary; McCabe advises choosing a path with as many decision nodes as possible. Next the baseline path is retraced, and in turn each decision is “flipped”, that is when a node of outdegree ≥2 is reached, a different edge must be taken. Apply it:

P1=(S,A,C,D,F); P2=(S,B,C,D,F), and P3=(S,B,C,E,F).

Solution to Task 1:

* 1. Control flow graph of sort1

9

1

2

3

4

5

6

7

8

10

11

12

13

14

15

16

17

* 1. Cyclomatic complexity M=E−N+P=17-13+1=5

Where E is the number of edges: E=17

N is the number of nodes: N=|{1..11, S, F}|=13

P is the number of the connected components: P=1

* 1. Paths are denoted by their nodes traversed (S, 1..11, and F). Nodes with out-degree>=2 (1, 3, 4, 5, 7) are used for flipping, Make first a baseline P1 having maximal possible number of flipping (conditional) nodes

P1: S,1,2,3,4,5,6,3,4,7,8,3,9,10,11,1,F

Then flip each conditional starting from the first one:

P2: S,1,F

P3: S,1,2,3,9,10,11,1,F

P4: S,1,2,3,4,5,3,4,7,8,3,9,10,11,1,F

P5: S,1,2,3,4,5,3,4,7,3,9,10,11,1,F

**Task 2. (5 points)** Give five reasons to use the code reverse engineering. Explain your answer

1. The code was developed long ago and developers are not available now
2. It is necessary to migrate to a new platform with another implementation language
3. Documentation is not available
4. Business rules changed and the software needs adaptation
5. Performance needs increasing

**Task 3. (6 points)** What is an E-type type program **(2 point)**? What is an anti-regressive activity **(2 points)** and why is it important for E-type programs evolution **(2 points)**?

E-type programs are program embedded into the real world so that they need satisfying everyday business requirements.

Anti-regressive activity aims making program code simpler, more efficient, and better understandable. It is important for E-type program because it simplifies understanding of the program and makes it more efficient.

Task 4. **(4 points)** What are the four software configuration management elements and what is the aim of each its element?

1. Identification of software configuration, i.e. defining the software product elements (configuration items) to be stored in the configuration database
2. Control of software configuration, i.e. defining formal procedures for check-in and check-out of the configuration items.
3. Auditing software configuration, i.e. defining procedures for checking correctness of the configuration
4. Accounting software configuration statue, i.e. recording history of configuration items alteration and providing access to it.