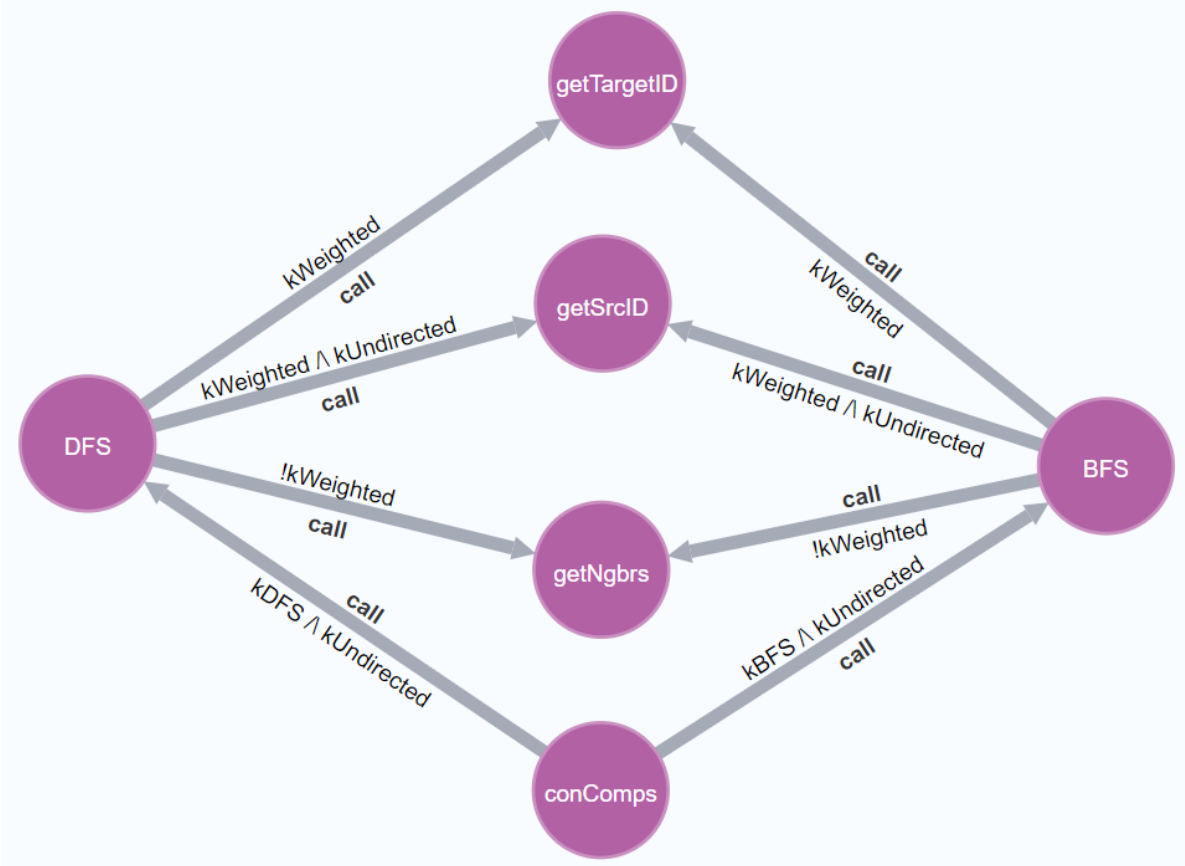
Introduction

In this questionnaire, you will be asked to perform 12 tasks that are similar to those that you performed at the first stage of the study. For these tasks, you will not have to run any query. Instead, we will provide you an image of the nodes and links for each task. Purple nodes represent functions and orange nodes represent variables from the code.

As in the previous stage of the study, you will be asked to identify links that represent program statements/actions that may execute in certain program variants. Recall that if a feature is not mentioned in a link's presence condition, then the feature's status (enabled or disabled) plays no role in whether the link can execute.

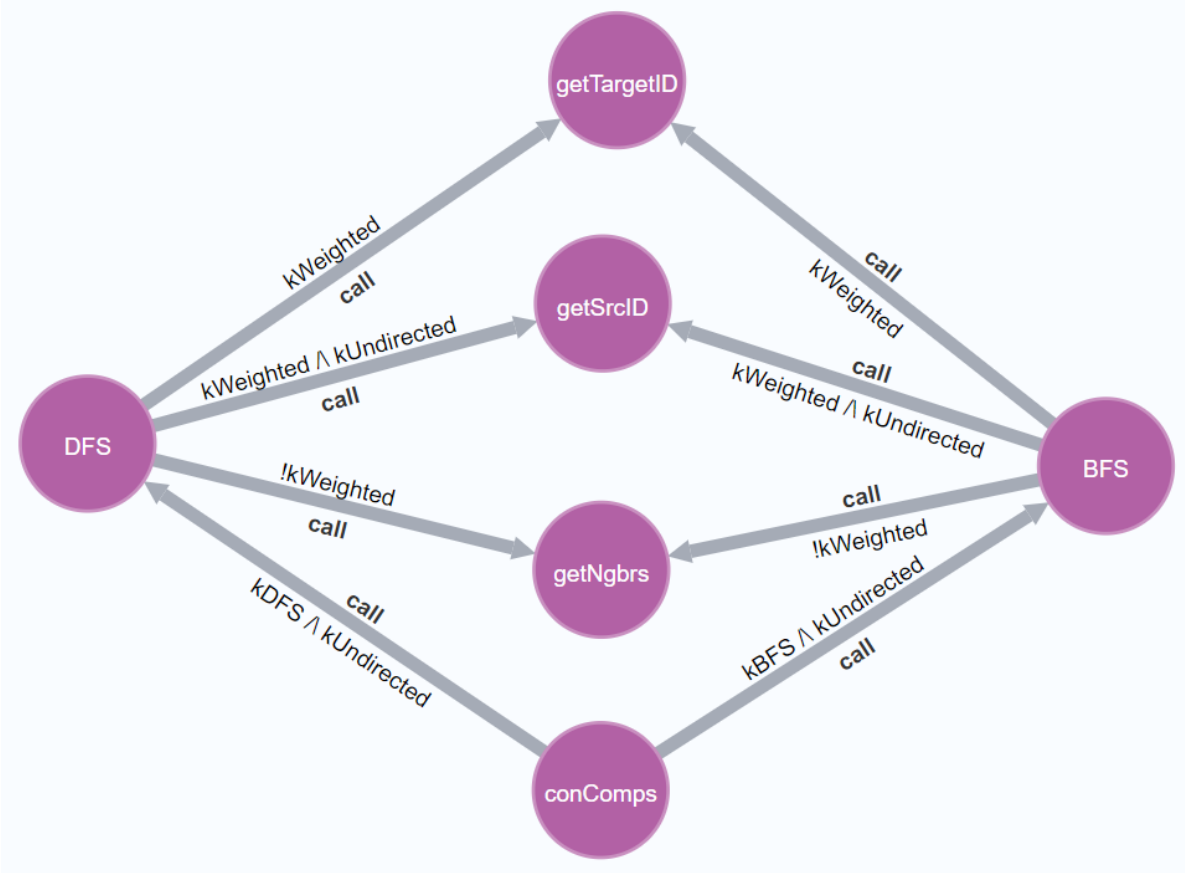
The bar at the top of the page shows your progress through the questionnaire.

Stage 2



List the name of the function(s) that may be called directly or indirectly by function

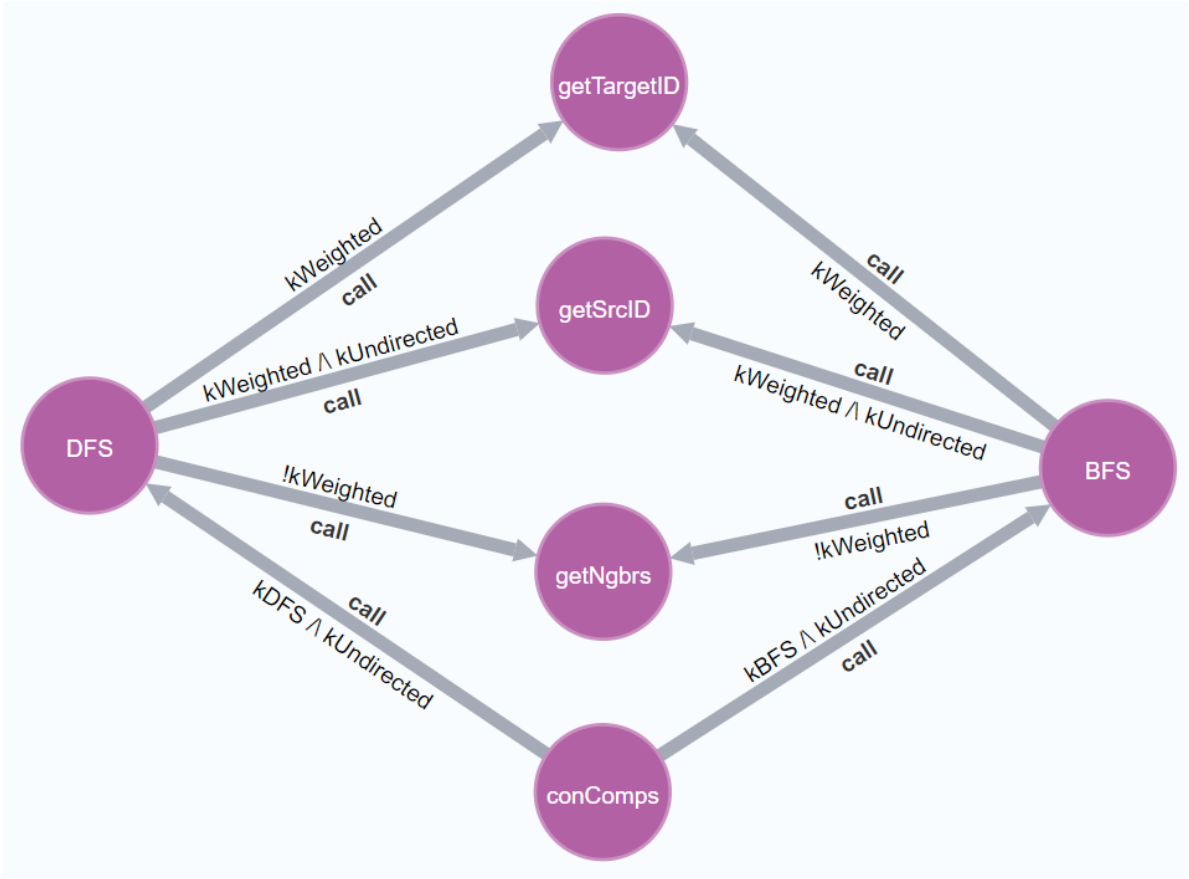
conComps in a variant with the following configuration: *kDFS /\ !kBFS /\ kUndirected /\ kWeighted*



V1: kDFS /\ !kBFS /\ kUndirected /\ kWeighted
V2: kDFS /\ !kBFS /\ kUndirected /\ !kWeighted

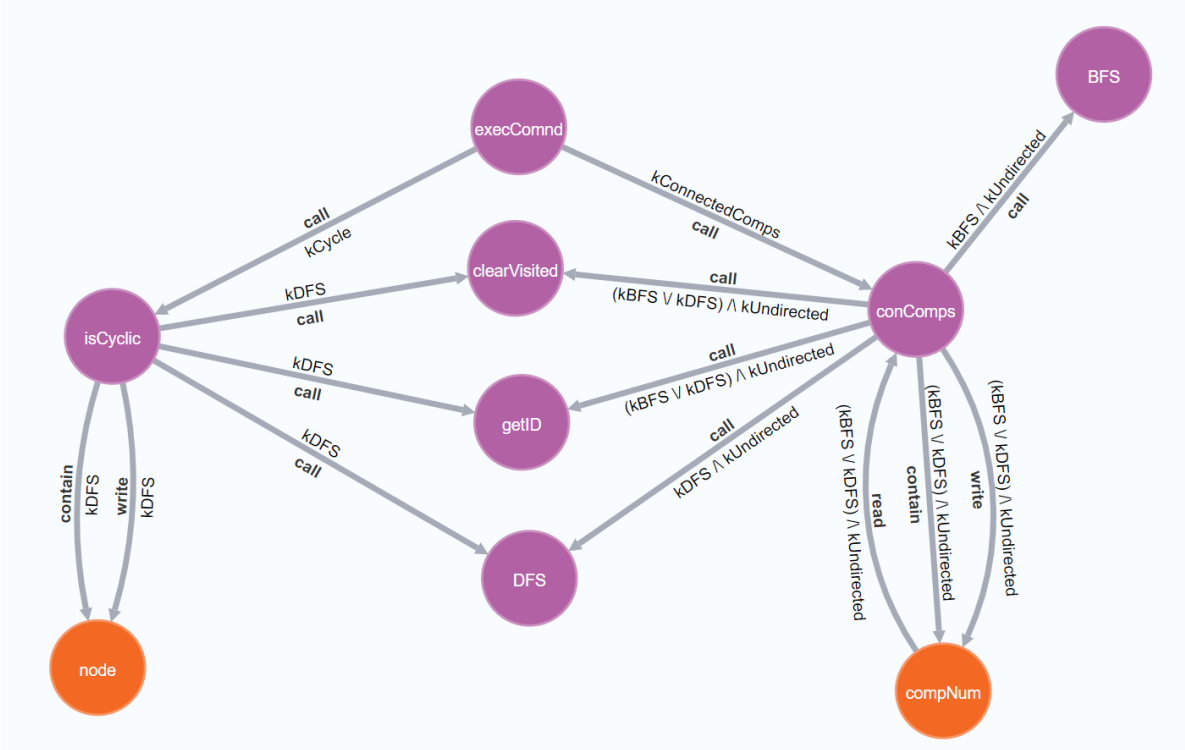
Which of the call chain(s) may execute in V1 but not in V2? Select all options that apply.

- [ ] conComps->DFS->getNgbrs
- [ ] conComps->DFS->getSrcID
- [ ] conComps->DFS->getTargetID



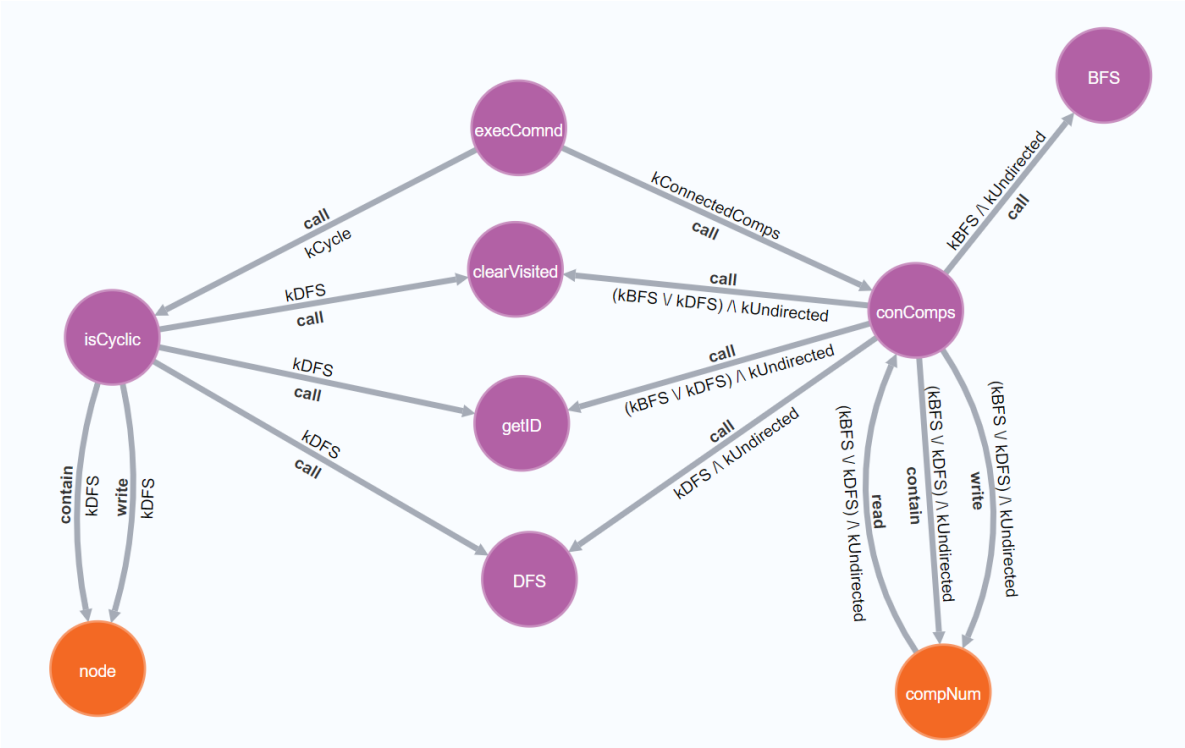
V1: kUndirected /\ kWeighted /\ kDFS /\ !kBFS
V2: !kUndirected /\ kWeighted /\ kDFS /\ !kBFS
V3: kUndirected /\ !kWeighted /\ !kDFS /\ kBFS

Considering the variants above, which variant does not execute any of the call chains starting at the **conComps** function?



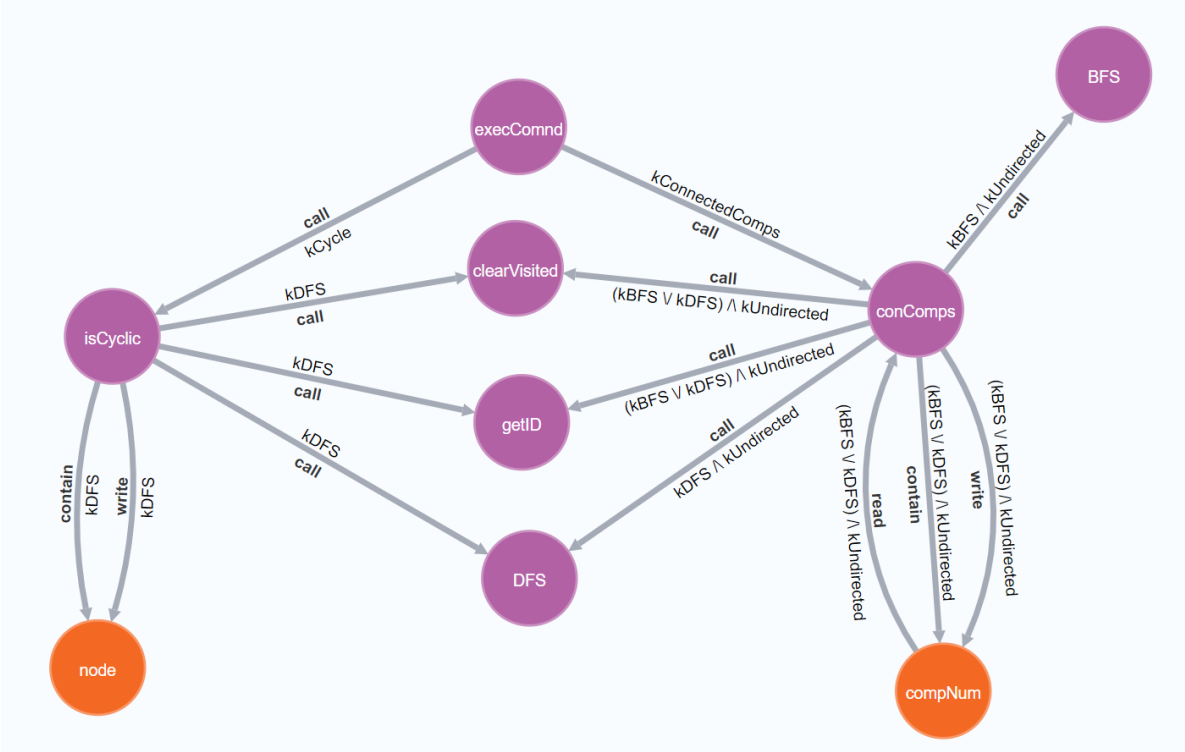
Which path(s) may fully execute in variant *kDFS /\ !kBFS /\ kUndirected /\ !kCycle /\ kConnectedComps*? Select all the options that apply.

- [ ] execComnd->isCyclic->getID
- [ ] execComnd->conComps->BFS
- [ ] execComnd->conComps->getID



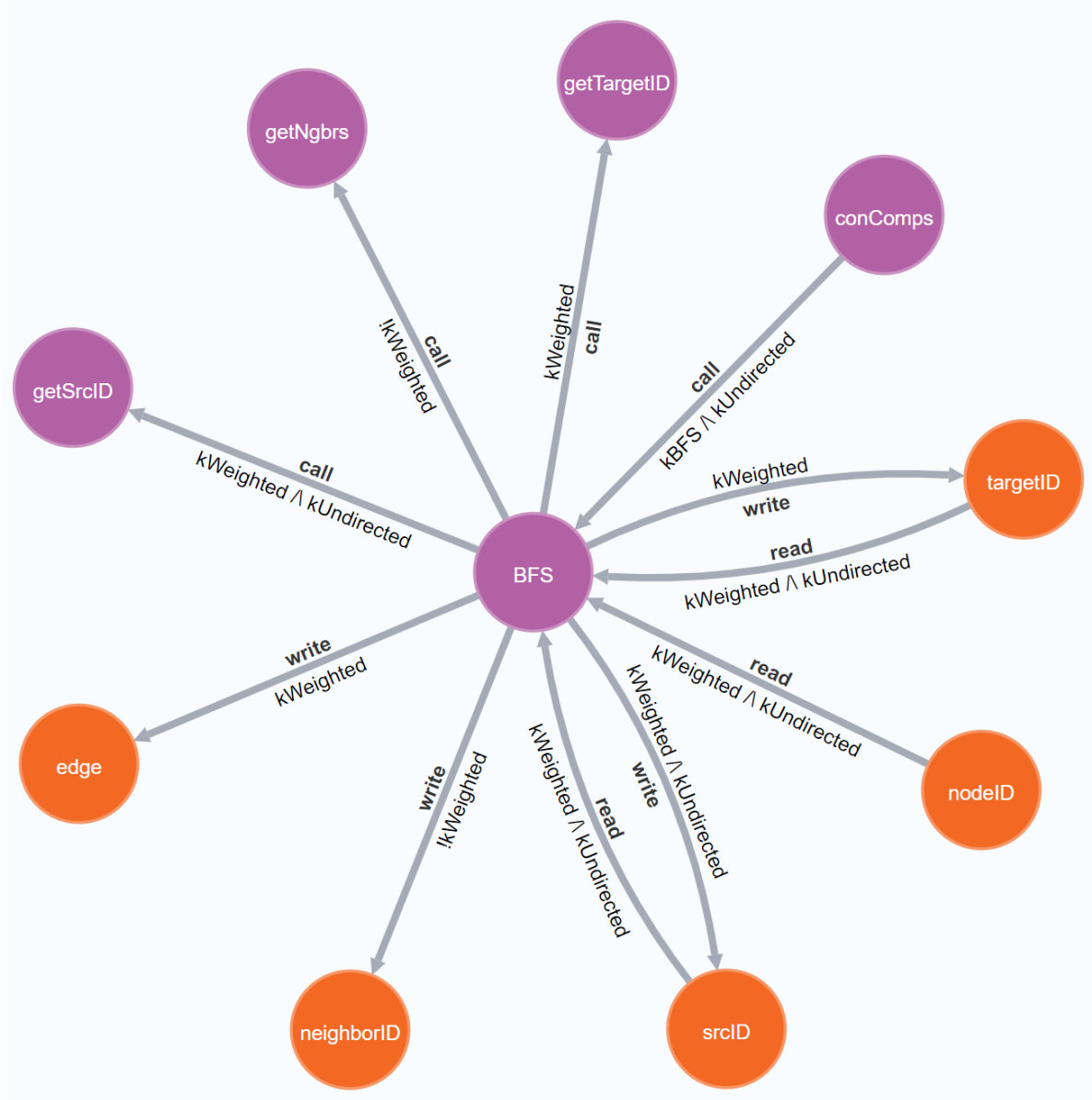
V1: !kUndirected /\ kConnectedComps /\ kBFS /\ !kDFS
V2: kUndirected /\ kConnectedComps /\ kBFS /\ !kDFS

Considering the variants above, which variant may execute the call path between functions **execCommand** and **BFS**?

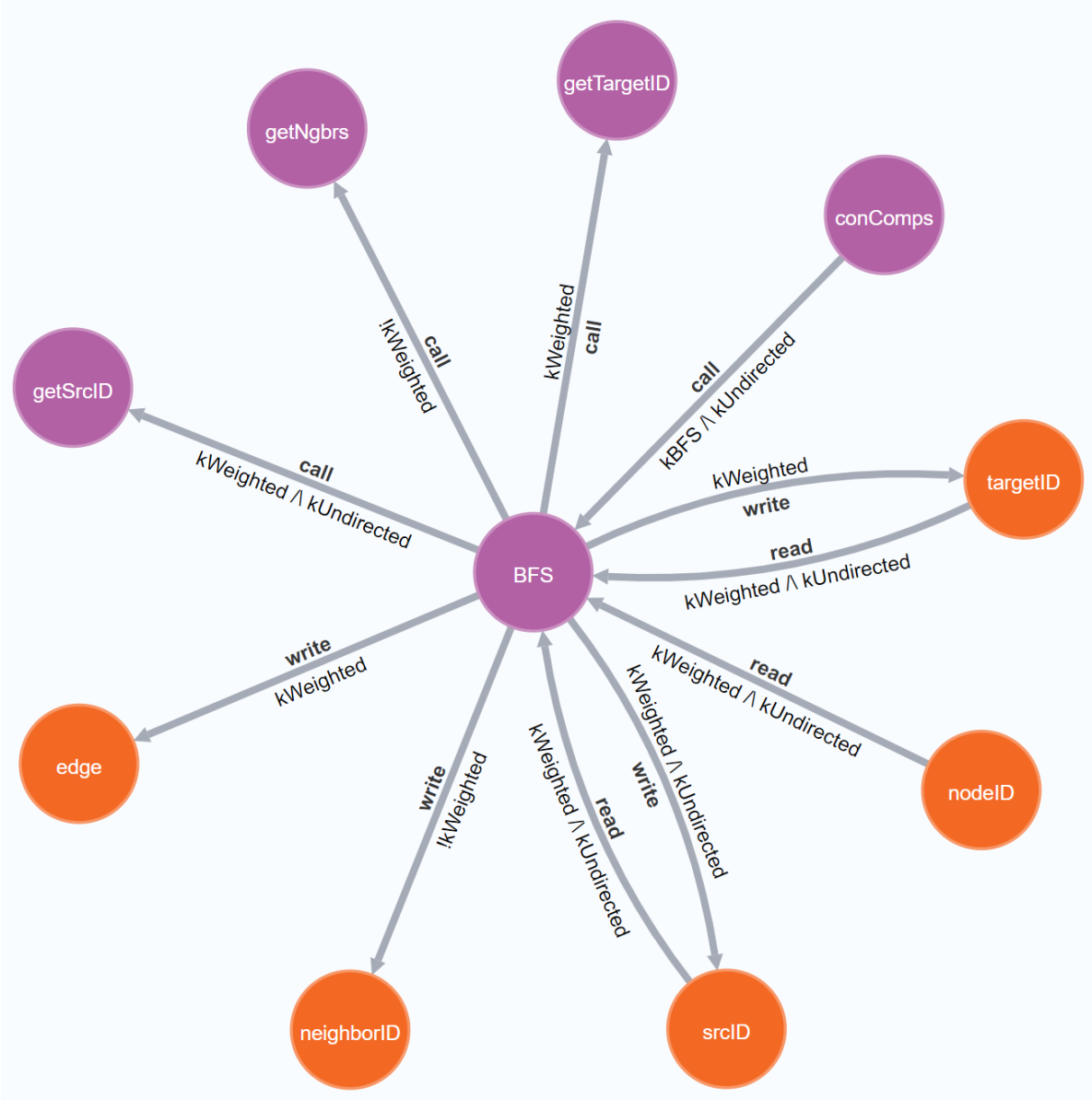


V1: kUndirected /\ kWeighted /\ kDFS /\ !kBFS
V2: kUndirected /\ !kWeighted /\ !kDFS /\ kBFS
V3: kUndirected /\ !kWeighted /\ kDFS /\ !kBFS

Considering the variants above, list the name of functions that may be called by **conComps** in either of the variants above, i.e., the listed functions may be called in V1, V2, and V3.

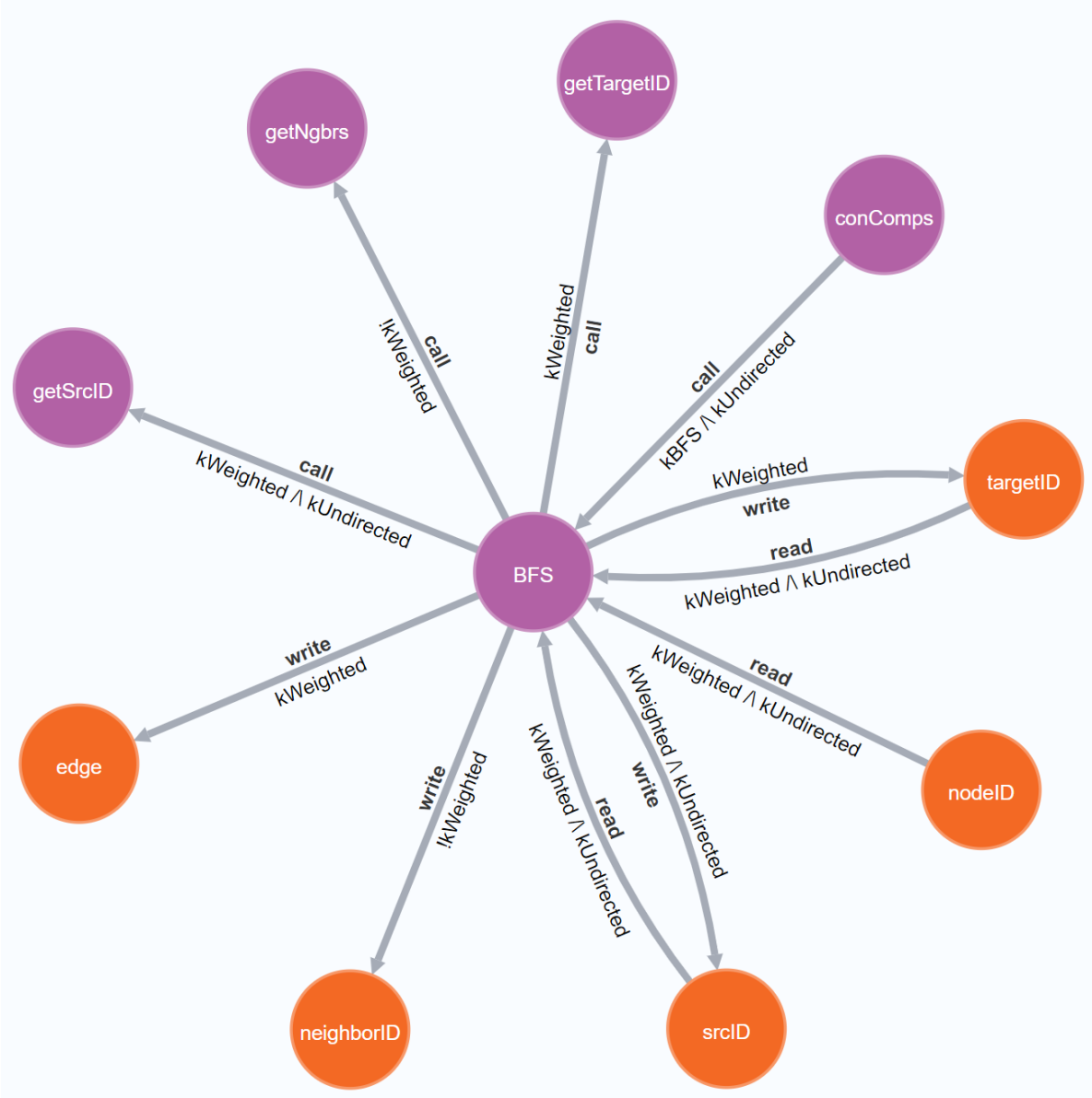


Which variable(s) may be written by **BFS** if the *kWeighted* feature is disabled?



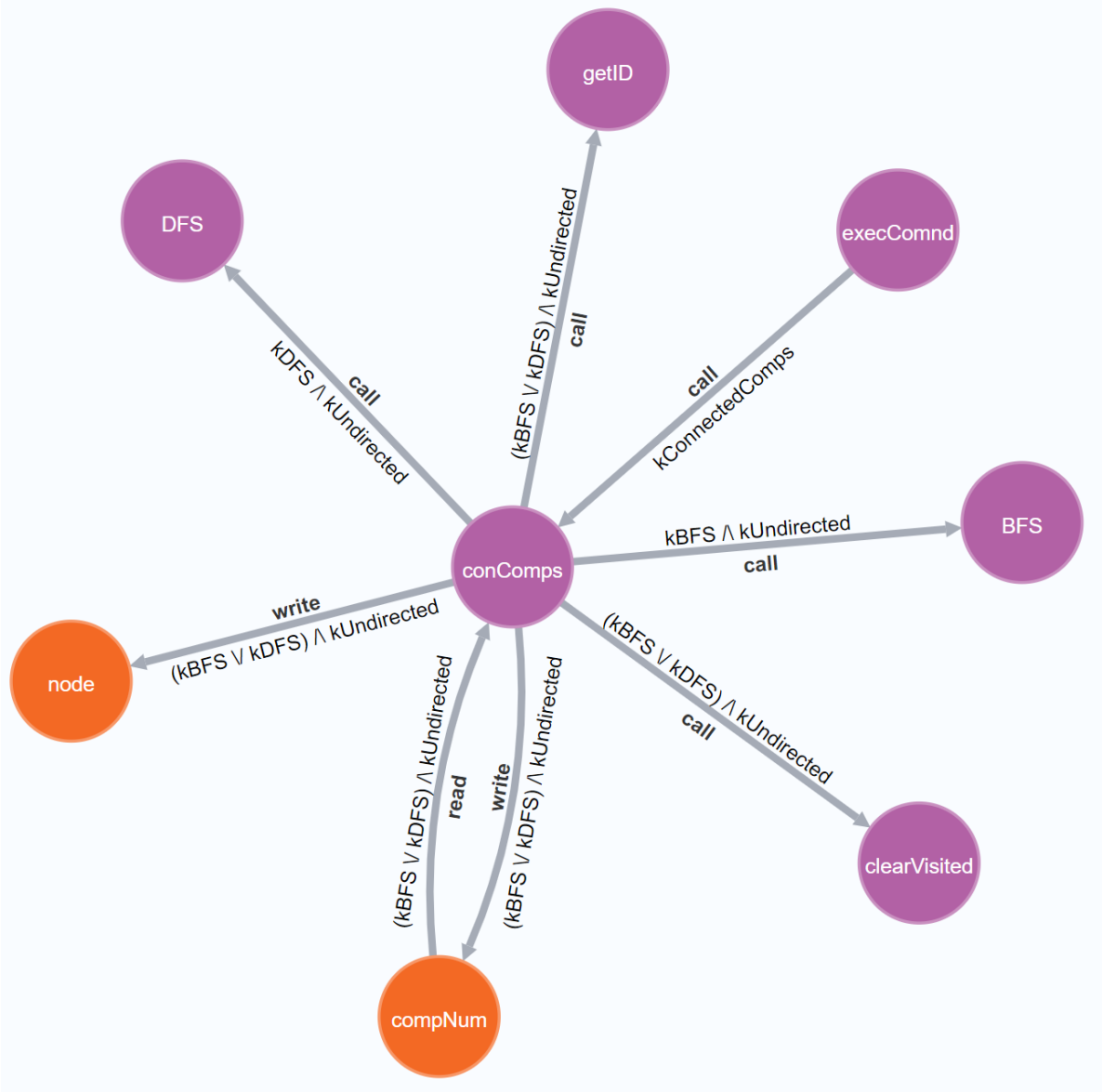
V1: kWeighted /\ !kUndirected
V2: kWeighted /\ kUndirected

In which variant the function **BFS** can read values from variables **nodeID, srcID, targetID**?

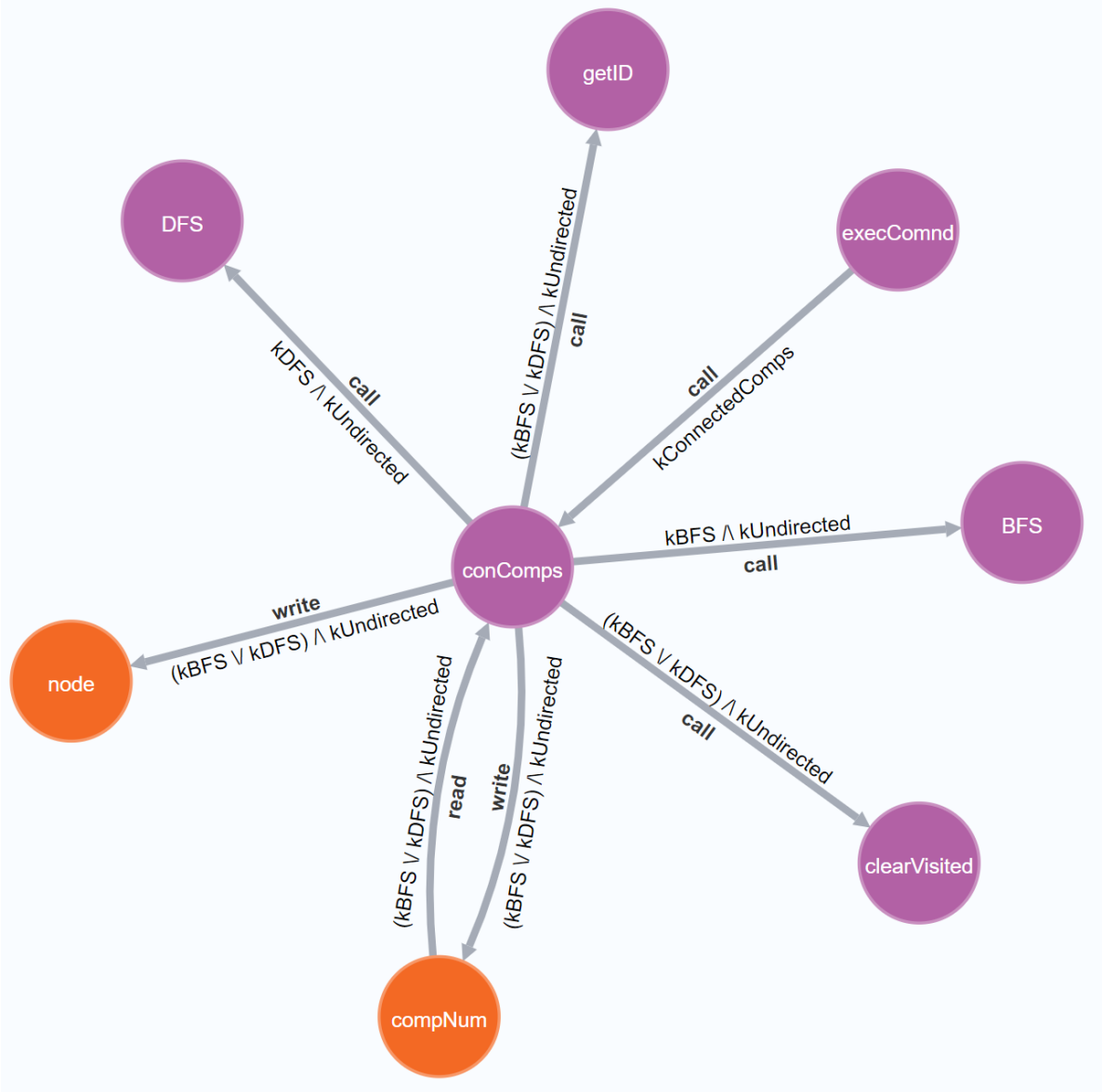


V1: kUndirected /\ kWeighted /\ !kBFS
V2: !kUndirected /\ kWeighted /\ !kBFS
V3: !kUndirected /\ !kWeighted /\ kBFS

Considering the variants above, in which variant does the function **BFS** have fewest relationships with other functions and variables? Please list the name of the nodes interacting with **BFS** in that variant and the type of their relationship.

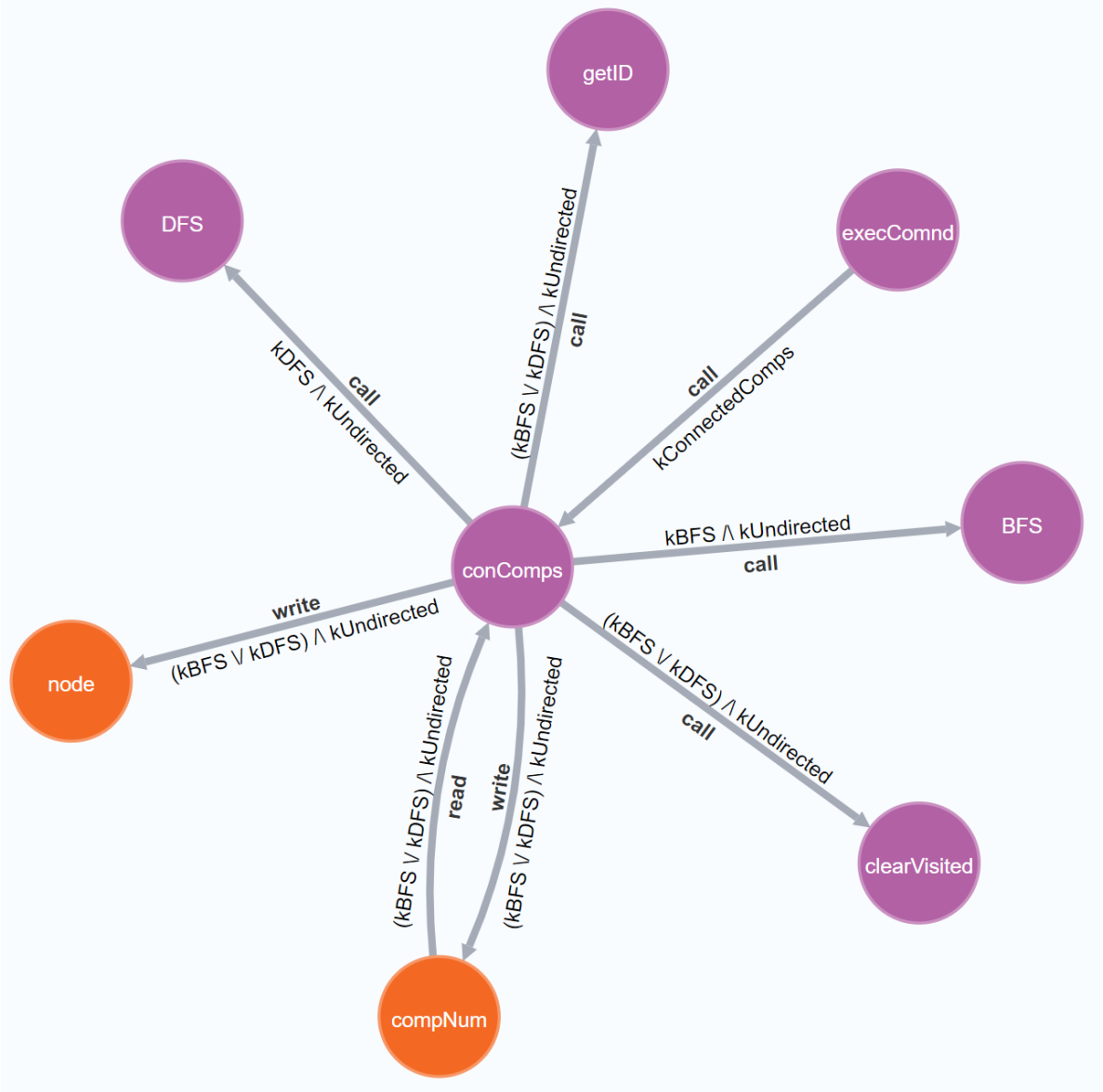


Is there any dataflow (read/write relationship) between the variable **node** and the function **conComps** in variant *kDFS /\ !kBFS /\ kUndirected*?



V1: kBFS /\ !kDFS /\ kUndirected
V2: !kBFS /\ kDFS /\ kUndirected

List the names of the functions that can be called by **conComps** in both variants



V1: kUndirected /\ kDFS /\ !kBFS /\ kConnectedComps
V2: kUndirected /\ !kDFS /\ kBFS /\ kConnectedComps
V3: !kUndirected /\ kDFS /\ !kBFS /\ kConnectedComps

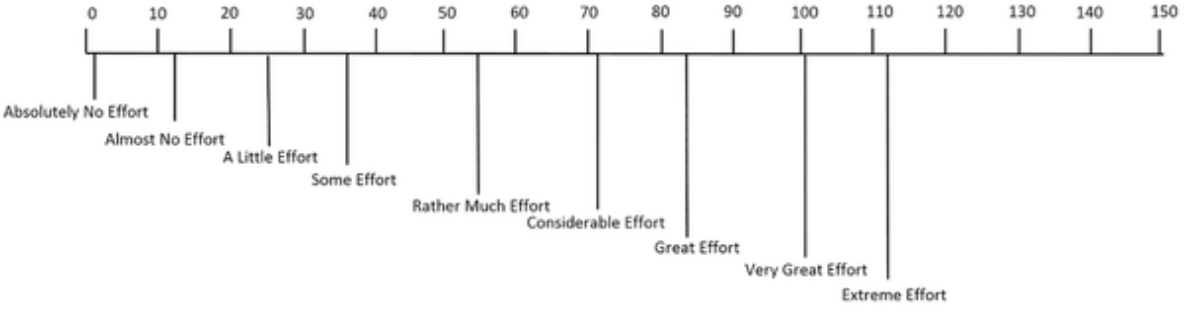
Which variant may **not** execute the following sequence: a call from function **execComnd** to function **conComps** followed by a write between function **conComps** and the variable **compNum**?

Feddback - no-filters

How much do you agree with the following statements?

	Strongly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Strongly agree
I found that interpreting the Boolean expressions were unnecessarily complex	○	○	○	○	○
I found the Boolean expressions very cumbersome to interpret	○	○	○	○	○
I found the Boolean expressions made it easier to identify the program variant in which a relationship holds	○	○	○	○	○
I needed to learn a lot of things before I could identify the program variant in which a relationship holds	○	○	○	○	○
I found the Boolean expressions were easy to access	○	○	○	○	○
I thought the graph customization was easy to use	○	○	○	○	○
I would imagine that most people would learn to customize the graph very quickly	○	○	○	○	○
I found the Boolean expressions increased the mental effort to perform the tasks	○	○	○	○	○

Place a mark on the scale indicating the mental effort you applied for each type of task



0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150

Find fact on program variant

Consider you have $100 dollars to spend on features that could improve the experience of identifying program facts that holds in particular program variants, the amount of dollars assigned to each feature represents its importance. Distribute your $100 dollars on the list below.

Customization of Boolean expression written on the link (e.g., font size)	$ 0
Automatic search of terms within Boolean expressions	$ 0
Multi-coloured links indicating the program variants in which they hold	$ 0
Hiding links that do not hold in a program variant	$ 0
Highlighting links that hold in a program variant	$ 0
Total	$ 0

What is your preferred syntax to represent boolean operations?

- ○ && (and), || (or)
- ○ /\ (and), \/ (or)
- ○ No preference. I am OK with either of them

Feedback and suggestion

What are your general impressions of the presented tool?

Do you have any suggestions for improvements to the presented tool? If so, please describe them below.