Lab 0 – Graph Creation & DFS Traversal

A) Write a program that will create a graph from a list of one-directional directions. The data file contains a list of one-directional connections from which you will construct the graph. You will use a map to store the graph. The toString method should show the contents of the map.

B) Write a recursive dfs method that will traverse the graph exactly as we did on the worksheet.

```
procedure DFS-recursive(v) is
    label v as discovered

    for all neighbors of v
        if neighbor is not labeled as discovered then
            recursively call DFS on the neighbor
```

C) Write an iterative dfs method that will traverse the graph exactly as the recursive version.

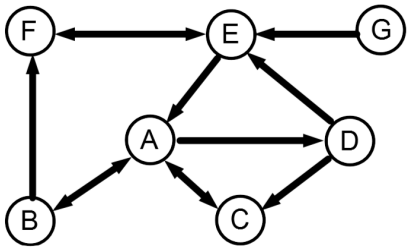
```
procedure DFS-iterative(v) is
   let visited be a stack
   push v onto stack
   while stack is not empty do
      print v
      v = stack.pop()
      if v is not labeled as discovered:
         label v as discovered
         loop thru v's list of neighbors*
            push each neighbor onto the stack
```

*If you loop FORWARD thru the neighbors, you'll get a valid but different DFS output than the recursive version. Loop BACKWARDS thru the neighbors to get the same output.

Graph 1's edges: `AB AC AD BA BF CA DC DE EA EF FE GE`

dfs(a): ABFECD

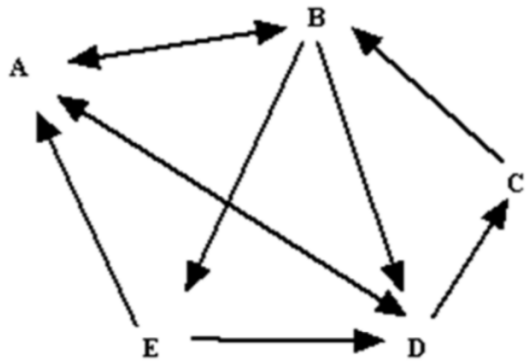
toString → `{A=BCD, B=AF, C=A, D=CE, E=AF, F=E, G=E}`



Graph 2's edges: `AB AD BA BD BE CB DA DC EA ED`

dfs(a): ABDCE

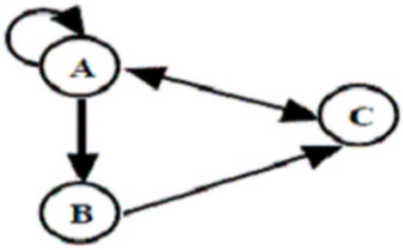
toString → `{A=BD, B=ADE, C=B, D=AC, E=AD}`



Graph 3's edges: `AA AB AC BC CA`

dfs(c): CAB

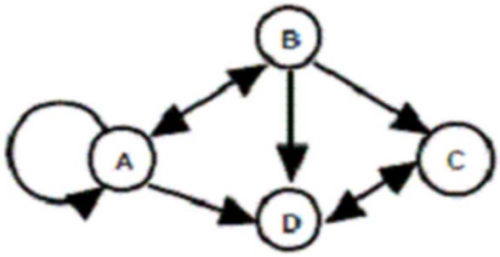
toString → `{A=ABC, B=C, C=A}`



Graph 4's edges: `AA AB AD BA BC BD CD DC`

dfs(d): DC

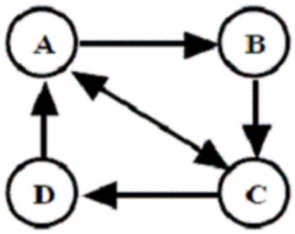
toString → `{A=ABD, B=ACD, C=D, D=C}`



Graph 5's edges: `AC DA CD CA BC AB`

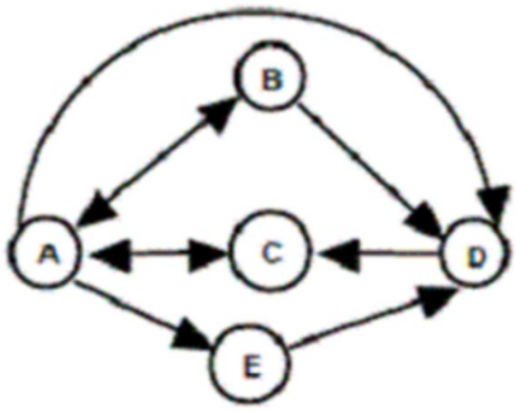
dfs(A): ABCD

toString → `{A=CB, B=C, C=DA, D=A}`



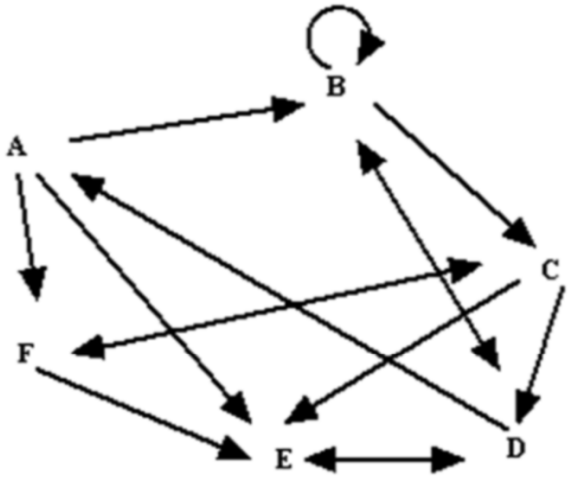
Graph 6's edges: BA AD CA ED AE AB AC DC BD **dfs(B): BADCE**

toString → `{A=DEBC, B=AD, C=A, D=C, E=D}`



Graph 7's edges: AB CD BB ED FC CE BC AE DA BD FE DB CF DE AF **dfs(E): EDABCF**

toString → `{A=BE, B=BCD, C=DEF, D=ABE, E=D, F=CE}`



Graph 8's edges: CG GC EF FE AC CA BA AB FB BF BD DB AE EA **dfs(A): ACGBFED**

toString → `{A=CB, B=AFD, C=GA, D=B, E=F, F=EB, G=C}`