Aayush Narayanan
naraya76@purdue.edu
CS 408 Project 1

1)

C/C++:

```
#include <stdio.h>

int main() {
   printf("%d\n", -5 % 2);
   return 0;
}
```

This returns -1

Java:

```
public class ModuloTest {
   public static void main(String[] args) {
      System.out.println(-5 % 2);
   }
}
```

This returns -1

Perl:

```
#!/usr/bin/perl
print -5 % 2, "\n";
```

This returns 1

Python:

```
print(-5 % 2)
```

This returns 1

Java, C/C++ return -1 and perl and python returns 1. This is because the first two use truncated division, whereas Perl and Python make the answer be the same sign as the divisor.

3 Strategies:
1. Standardization: Standardize how modulo is performed across different languages
2. Teach Developers the differences when using modulo in different languages or environments
3. Custom Functions: Developers can make their own functions that make the modulo work the same across different languages

2)

a)

```
==1803959== Memcheck, a memory error detector
==1803959== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1803959== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1803959== Command: ./sll_fixed
==1803959==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>100
enter the name:>Tom
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>111
enter the name:>Mary
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:d
enter the tel :>111
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:x
bye
==1803959==
==1803959== HEAP SUMMARY:
==1803959==     in use at exit: 9 bytes in 1 blocks
==1803959==   total heap usage: 7 allocs, 6 frees, 2,115 bytes allocated
==1803959==
==1803959== 9 bytes in 1 blocks are definitely lost in loss record 1 of 1
==1803959==    at 0x484DCD3: realloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd6
==1803959==    by 0x1093A9: fgets_enhanced (in /u/riker/u92/naraya76/sll_fixed)
==1803959==    by 0x109A60: main (in /u/riker/u92/naraya76/sll_fixed)
==1803959==
==1803959== LEAK SUMMARY:
==1803959==    definitely lost: 9 bytes in 1 blocks
==1803959==    indirectly lost: 0 bytes in 0 blocks
==1803959==      possibly lost: 0 bytes in 0 blocks
==1803959==    still reachable: 0 bytes in 0 blocks
==1803959==         suppressed: 0 bytes in 0 blocks
==1803959==
==1803959== For lists of detected and suppressed errors, rerun with: -s
==1803959== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
naraya76@data:~$ vim
```

When running valgrind for test case 1, the output is a memory leak of 9 bytes.This because the memory is not properly allocated when realloc is called in the fgets_enhanced function. To fix this change, I changed the delete_node function. In the function, I did free(temp->str); before free(temp). This fixed the issue as it made sure to release the memory before exiting. Output of fixed:

```
naraya76@data:~$ valgrind --leak-check=full ./sll_fixed
==1928707== Memcheck, a memory error detector
==1928707== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1928707== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1928707== Command: ./sll_fixed
==1928707==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>100
enter the name:>Tom
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>111
enter the name:>Mary
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:d
enter the tel :>111
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:x
bye
==1928707==
==1928707== HEAP SUMMARY:
==1928707==     in use at exit: 0 bytes in 0 blocks
==1928707==   total heap usage: 7 allocs, 7 frees, 2,115 bytes allocated
==1928707==
==1928707== All heap blocks were freed -- no leaks are possible
==1928707==
==1928707== For lists of detected and suppressed errors, rerun with: -s
==1928707== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
naraya76@data:~$
```

b)

```
naraya76@data:~$ valgrind --leak-check=full ./sll_buggy
==2084394== Memcheck, a memory error detector
==2084394== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2084394== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2084394== Command: ./sll_buggy
==2084394==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>100
enter the name:>Tom
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>111
enter the name:>Mary
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>112
enter the name:>John
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:e
enter the old tel :>111
enter the new tel :>111
enter the new name:>Mary
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:a
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:x
==2084394== Invalid read of size 8
==2084394==    at 0x109522: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109BD0: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Address 0x4ab5920 is 16 bytes inside a block of size 24 free'd
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109544: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109B87: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Block was alloc'd at
==2084394==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x10982E: append (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109A72: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==
==2084394== Invalid read of size 8
==2084394==    at 0x10952E: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109BD0: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Address 0x4ab5910 is 0 bytes inside a block of size 24 free'd
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109544: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109B87: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Block was alloc'd at
==2084394==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x10982E: append (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109A72: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==
==2084394== Invalid free() / delete / delete[] / realloc()
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109538: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109BD0: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Address 0x4ab58c0 is 0 bytes inside a block of size 5 free'd
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109538: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109B87: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Block was alloc'd at
==2084394==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109315: fgets_enhanced (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109A60: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==
==2084394== Invalid free() / delete / delete[] / realloc()
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109544: delete_all (in /u/riker/u92/naraya76/sll_buggy)
```

```
==2084394== Invalid free() / delete / delete[] / realloc()
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109544: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109BD0: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Address 0x4ab5910 is 0 bytes inside a block of size 24 free'd
==2084394==    at 0x484B27F: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x109544: delete_all (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109B87: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==  Block was alloc'd at
==2084394==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2084394==    by 0x10982E: append (in /u/riker/u92/naraya76/sll_buggy)
==2084394==    by 0x109A72: main (in /u/riker/u92/naraya76/sll_buggy)
==2084394==
bye
==2084394==
==2084394== HEAP SUMMARY:
==2084394==     in use at exit: 0 bytes in 0 blocks
==2084394==   total heap usage: 12 allocs, 18 frees, 2,167 bytes allocated
==2084394==
==2084394== All heap blocks were freed -- no leaks are possible
==2084394==
==2084394== For lists of detected and suppressed errors, rerun with: -s
==2084394== ERROR SUMMARY: 12 errors from 4 contexts (suppressed: 0 from 0)
naraya76@data:~$
```

This is the output of running Valgrind for test case 2, in the output shown, there are a couple of problems. First there is an invalid read of size 8, and then an invalid free/delete error. The first error happens when memory that's already been freed is tried to be read, and the second happens when memory that's previously been freed is being attempted to be freed again.To fix these errors, we must add a fix in the delete_all function that will make sure the pointer is not null when we call free, so we set p to null. Output of running fixed:

```
naraya76@data:~$ valgrind --leak-check=full ./sll_fixed
==2502973== Memcheck, a memory error detector
==2502973== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2502973== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2502973== Command: ./sll_fixed
==2502973==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>100
enter the name:>Tom
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>111
enter the name:>Mary
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>112
enter the name:>John
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:e
enter the old tel :>111
enter the new tel :>111
enter the new name:>Mary
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:a
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:x
bye
==2502973==
==2502973== HEAP SUMMARY:
==2502973==     in use at exit: 0 bytes in 0 blocks
==2502973==   total heap usage: 12 allocs, 12 frees, 2,167 bytes allocated
==2502973==
==2502973== All heap blocks were freed -- no leaks are possible
==2502973==
==2502973== For lists of detected and suppressed errors, rerun with: -s
==2502973== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

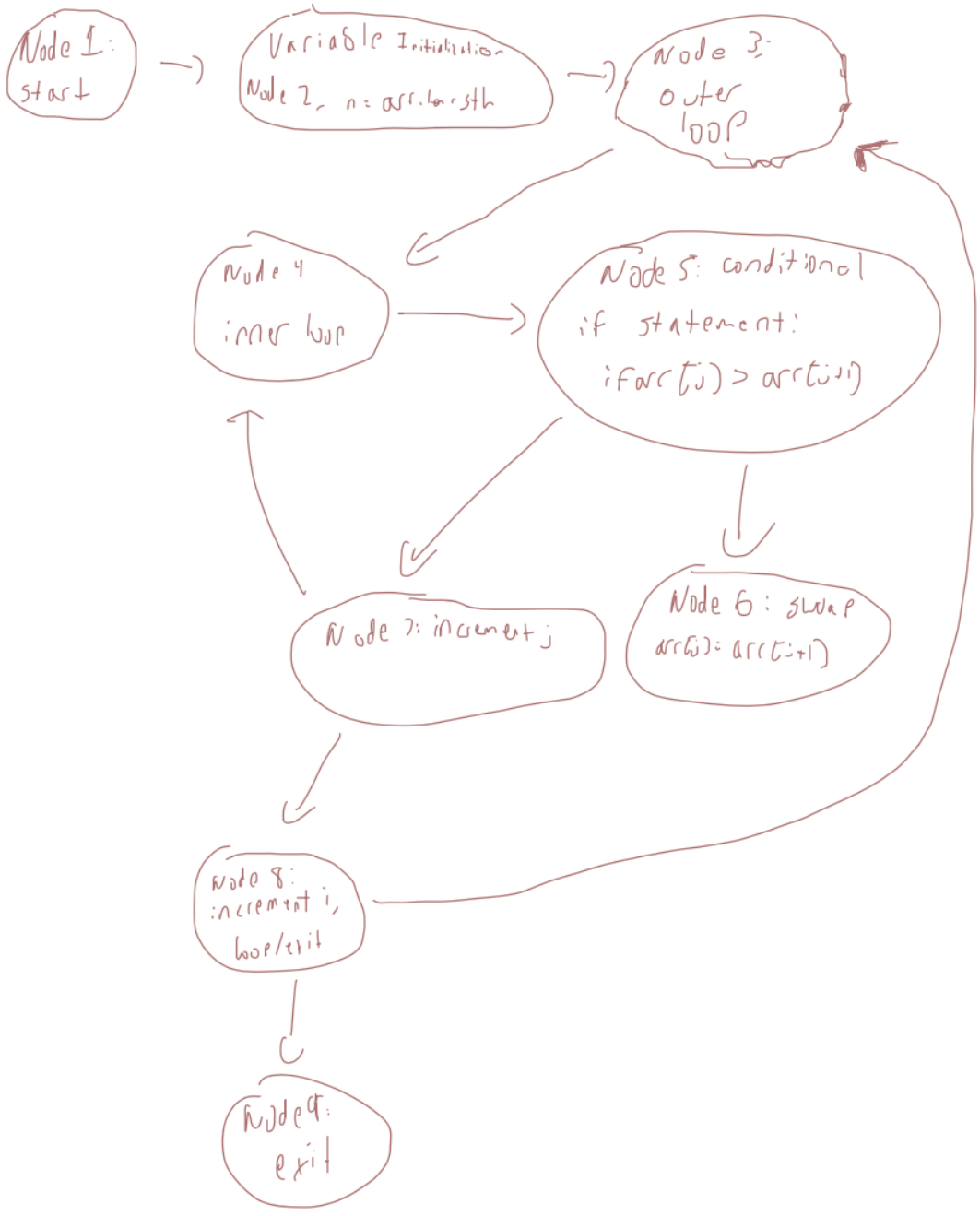
c)

```
naraya76@data:~$ valgrind --leak-check=full ./sll_buggy
==2103949== Memcheck, a memory error detector
==2103949== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2103949== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2103949== Command: ./sll_buggy
==2103949==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>123
enter the name:>Aayush
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:u
enter the tel :>123
==2103949== Invalid write of size 1
==2103949==    at 0x484EE8E: strcpy (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2103949==    by 0x1097BD: duplicate (in /u/riker/u92/naraya76/sll_buggy)
==2103949==    by 0x109BC4: main (in /u/riker/u92/naraya76/sll_buggy)
==2103949==  Address 0x4ab59c6 is 0 bytes after a block of size 6 alloc'd
==2103949==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2103949==    by 0x1097A3: duplicate (in /u/riker/u92/naraya76/sll_buggy)
==2103949==    by 0x109BC4: main (in /u/riker/u92/naraya76/sll_buggy)
==2103949==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:p
==2103949== Invalid read of size 1
==2103949==    at 0x484ED24: strlen (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2103949==    by 0x48FFD30: __vfprintf_internal (vfprintf-internal.c:1517)
==2103949==    by 0x48E979E: printf (printf.c:33)
==2103949==    by 0x109924: display (in /u/riker/u92/naraya76/sll_buggy)
==2103949==    by 0x109B7B: main (in /u/riker/u92/naraya76/sll_buggy)
==2103949==  Address 0x4ab59c6 is 0 bytes after a block of size 6 alloc'd
==2103949==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2103949==    by 0x1097A3: duplicate (in /u/riker/u92/naraya76/sll_buggy)
==2103949==    by 0x109BC4: main (in /u/riker/u92/naraya76/sll_buggy)
==2103949==
The elements are :>  -> 123, Aayush -> 123, Aayush
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:x
bye
==2103949==
==2103949== HEAP SUMMARY:
==2103949==     in use at exit: 0 bytes in 0 blocks
==2103949==   total heap usage: 7 allocs, 7 frees, 2,116 bytes allocated
==2103949==
==2103949== All heap blocks were freed -- no leaks are possible
==2103949==
==2103949== For lists of detected and suppressed errors, rerun with: -s
==2103949== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

Here, you can see there are 2 errors at the bottom. This is due to the fact that we have to add one to the malloc size so that the null terminator is accounted for. Output for 2c of sll_fixed:

```
naraya76@data:~$ valgrind --leak-check=full ./sll_fixed
==2214868== Memcheck, a memory error detector
==2214868== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2214868== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==2214868== Command: ./sll_fixed
==2214868==
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:i
enter the tel:>123
enter the name:>Aayush
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:u
enter the tel :>123
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:p
The elements are :>  -> 123, Aayush -> 123, Aayush
[(i)nsert,(d)elete,delete (a)ll,d(u)plicate,(e)dit,(p)rint,e(x)it]:x
bye
==2214868==
==2214868== HEAP SUMMARY:
==2214868==     in use at exit: 0 bytes in 0 blocks
==2214868==   total heap usage: 7 allocs, 7 frees, 2,117 bytes allocated
==2214868==
==2214868== All heap blocks were freed -- no leaks are possible
==2214868==
==2214868== For lists of detected and suppressed errors, rerun with: -s
==2214868== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Q3) CFG:


Node 1:
start
Variable Initialization
Node 2, n = arr.length
Node 3:
outer
loop
Node 4
inner loop
Node 5: conditional
if statement:
if arr[j] > arr[j+1]
Node 6: swap
arr[j] = arr[j+1]
Node 7: increment j
Node 8:
increment i,
loop/exit
Node 9:
exit

4a)
Each node must be executed at least once
TR_{NC}: [N1], [N2], [N3], [N4], [N5], [N6], [N7], [N8], [N9], [N10], [N11]

TR_{EC}: Each edge must be used once
[N1 → N2]
[N1 → N3]
[N2 → N3]
[N3 → N4]
[N3 → N5]
[N3 → N6]
[N3 → N7]
[N4 → N8]
[N5 → N8]
[N6 → N7]
[N7 → N8]
[N8 → N9]
[N8 → N10]
[N9 → N11]
[N10 → N11]

TR_{EPC}: All pairs of consecutive edges must be covered for EPC.
[N1 → N2 → N3]
[N1 → N3 → N4]
[N1 → N3 → N5]
[N1 → N3 → N6]
[N1 → N3 → N7]
[N2 → N3 → N5]
[N2 → N3 → N6]
[N2 → N3 → N7]
[N3 → N4 → N8]
[N3 → N5 → N8]
[N3 → N6 → N7]
[N3 → N7 → N8]
[N4 → N8 → N9]
[N4 → N8 → N10]
[N5 → N8 → N9]
[N5 → N8 → N10]
[N6 → N7 → N8]
[N7 → N8 → N9]
[N7 → N8 → N10]
[N8 → N9 → N11]
[N8 → N10 → N11]

TR_{PPC}: For all prime paths to be covered, test requirements are:
TRppc (Reachable):
[N1 → N2 → N3 → N6 → N7 → N8 → N9 → N11]
[N1 → N2 → N3 → N5 → N8 → N9 → N11]
[N1 → N2 → N3 → N4 → N8 → N10 → N11]
[N1 → N2 → N3 → N7 → N8 → N9 → N11]
[N1 → N3 → N6 → N7 → N8 → N9 → N11]
[N1 → N3 → N4 → N8 → N10 → N11]
TRppc (Unreachable):
[N1 → N2 → N3 → N6 → N7 → N8 → N10 → N11]
[N1 → N2 → N3 → N5 → N8 → N10 → N11]
[N1 → N2 → N3 → N4 → N8 → N9 → N11]
[N1 → N3 → N6 → N7 → N8 → N10 → N11]
[N1 → N2 → N3 → N7 → N8 → N10 → N11]
[N1 → N3 → N4 → N9 → N11]
[N1 → N3 → N5 → N8 → N10 → N11]
[N1 → N3 → N7 → N8 → N10 → N11]

5)
a)
addNode test case has the test path 1,2,3 and it visits all three nodes in the CFG. The addNode_duplicate test has the test path 1,3. With this second path, all edges are visited in the CFG as well. Therefore it passes both edge and node coverage.

b)
addEdge covers each node(1,2,3,4,5,6,7) and therefore it passes node coverage. However, after the test paths are done, it is still missing some edges, (5,7) and (3,5) so therefore edge coverage is not passed.

c)
deleteNode has test paths that cover all edges and nodes in the CFG so therefore it passes both node and edge coverage. For edge coverage, the paths (1, 2, 3, 4, 3, 5) and (1, 5) from the deleteNode_missing test case together cover all edges in the graph

d)
the deleteEdge test case has the test path 1,2,3 and it visits all three nodes in the CFG. The deleteEdge_missing has the test path 1,3. With this second path, all edges are visited in the CFG as well. Therefore it passes both edge and node coverage.

5f)

addEdge is the only one that doesn't meet the requirements for both edge and node coverage. This because it is missing two edges for edge coverage, it however does go through all nodes.

It fails to go through 3,5 and 5,7. DeleteNode, addNode, and deleteEdge all go through each node and edge and therefore pass both node and edge coverage. Graphs below.