COEN 448 Software Testing and Validation Assignment 2 2023

White Box Testing

In this assignment, we aim to practise white box testing and coverage strategies to produce unit test cases.

Problem 1 (30 MARKS) The UserInputHandler program code is provided in the attachment. It is a simple function that stores the user's input until a certain terminating token is typed. The original code applies the token 'bye' to terminate. Now please revise the program to make the options with token 'bye' or 'quit' or 'exit'.

1.1. Develop CFG of the revised program and compute the CC value (5 MARKS).
1.2. Based on the CC value, derive the basic paths. (5 MARKS)
1.3. Analyze the program's code structure and estimate the number of test cases needed for condition coverage and decision coverage respectively. (5MARKS)
1.4. Generate test cases for 100% basic path coverage, condition coverage and decision coverage. Please make table for teach type of coverage. (15 MARKS)

5MARKS	Basic path
Test case t1{REPLACE HERE THE TEST CASE}	{REPLACE HERE TO PATH}

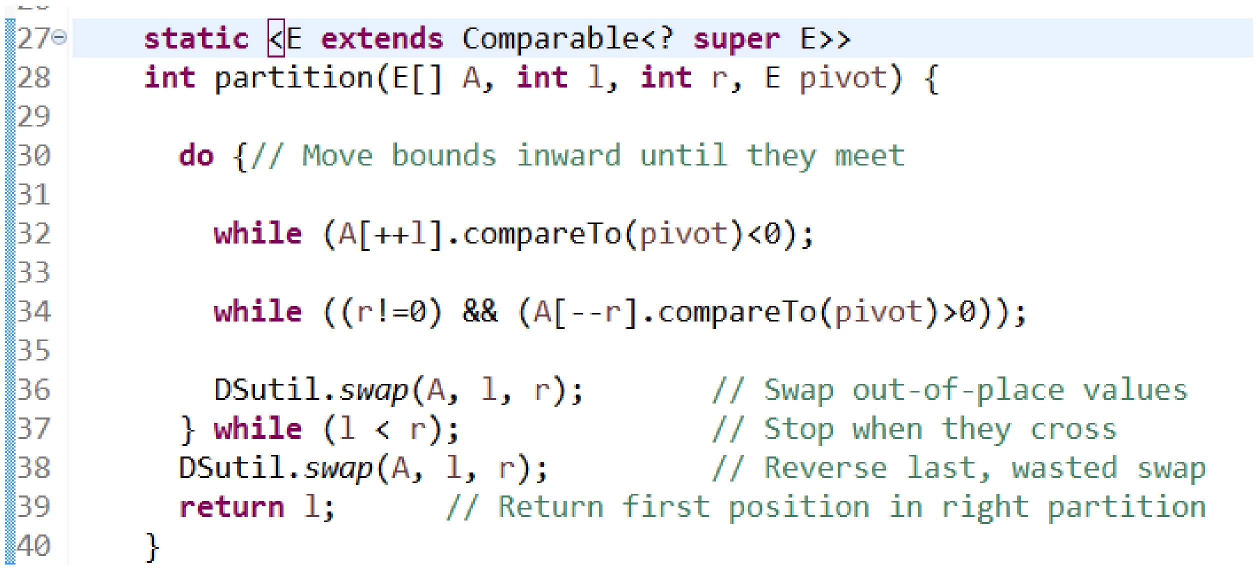
(5MARKS)	Condition
Test case t1{REPLACE HERE THE TEST CASE}	{CONDITION EXPRESS == TURE}
Test case t2{REPLACE HERE THE TEST CASE}	{CONDITION EXPRESS == FALSE}

(5MARKS)	Decision
Test case ti{REPLACE HERE THE TEST CASE}	{DECISION EXPRESS == TURE}
Test case tj{REPLACE HERE THE TEST CASE}	{DECISION EXPRESS == FALSE}

Problem 2 (45 Marks): The QuickSort algorithm code is provided in the attachment of this assignment. In QuickSort, the pivot is an element in the array. The function partition() returns the position of the pivot in the array so that the left side of the pivot has elements smaller to the value of the pivot while the right side of the pivot has the elements equal or bigger to the value of the pivot.

2.1 **(10 Marks) Develop test cases** following the black-box approach that has input domain modeling of the partition function according to the (2.1.a) worse case and (2.1.b) average case of the quick sort algorithm. You can choose base choice coverage or other coverage criterion to develop the test cases.

2.2 **(15 Marks) Write the unite test cases** for (2.1.a) and (2.1.b) , run those test cases and produce the coverage report from your programming IDE.



```java
    static <E extends Comparable<? super E>>
    int partition(E[] A, int l, int r, E pivot) {

      do {// Move bounds inward until they meet

        while (A[++l].compareTo(pivot)<0);

        while ((r!=0) && (A[--r].compareTo(pivot)>0));

        DSutil.swap(A, l, r);       // Swap out-of-place values
      } while (l < r);              // Stop when they cross
      DSutil.swap(A, l, r);         // Reverse last, wasted swap
      return l;      // Return first position in right partition
    }
```

2.3 (20 Marks)(2.3.1)Produce a CFG of the partition function. Leverage the table based def-use pair approach, and produce the table below for variable ***pivot.***

node i	def(i)	c-use(i)	edge(i,j)	p-use(i,j)

node i	dcu(v,i)	dpu(v,i)

(2.3.2) Develop test cases to cover all the dcu and dpu.

(2.3.3) Program unit test cases, run the test cases and produce a coverage report from you IDE.

Submission:

1. A report document that writes the following section.
    - **Solutions** to Problem 1 and Problem 2 including tables, test cases, CFG and screenshots of the coverage reports.
    - Observe the coverage reports of Problem 2. **Discussion** of the difference in the coverage and explain the possible cause of the difference.
    - **Comments** on the pros and cons of the two testing approaches (structural testing and data flow testing).
2. The unit test source code.

Please pack all the submission in one archive file with .zip or .tar. No .rar files are acceptable.