Question #1: Branch Software Testing (25%)

**Generate a program flow graph based on the source code of DoSelectionSort() function. (10%)**

void quicksort (element list[], int left, int right)

{

int pivot, i, j;

element temp;

1. if (left < right)

{

2. i = left; j = right+1;

3. pivot = list[left].key;

4. do

{

5. do i++;

6. while (list[i].key < pivot);

8 do j--;

9 while (list[j].key > pivot);

10 if (i < j)

11. SWAP(list[i], list[j], temp);

}

12 while (i < j);

13 SWAP(list[left], list[j], temp);

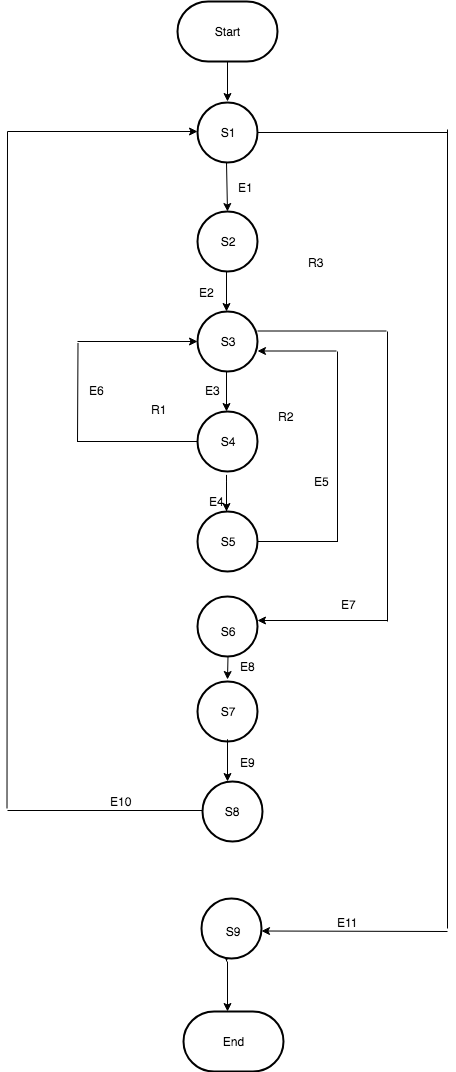
14 quicksort(list, left, j-1);

15 quicksort(list, j+1, right);

}

}

a) Generate a program flow graph based on the source code of DoSelectionSort() function. (10%)



b) Branch table for Selection sort branch testing.

|  |  |  |
| --- | --- | --- |
| Predicate Node | Decision | Outcome |
| S1 | i < arr.length - 1 | F |
|  |  | T |
| S3 | j < arr.length | T |
|  |  | F |
| S4 | arr[j] < arr[index | T |
|  |  | F |

|  |  |  |  |
| --- | --- | --- | --- |
| Predicate Node | Decision | Outcome | Path |
| S1 | i < arr.length - 1 | T | P2,P3,P4 |
|  |  | F | P1 |
| S3 | j < arr.length | T | P3,P4 |
|  |  | F | P2 |
| S4 | arr[j] < arr[index] | T | P4 |
|  |  | F | P3 |

P1 : S1 -> S9

P2 : S1-> S2 ->S3 ->S6 -> S7 -> S8 -> S1 ->S9

P3: S1-> S2 ->S3 ->S4 -> S3-> S6-> S7 -> S8 -> S1 ->S9

P4 : S1-> S2 ->S3 ->S4 -> S5 -> S3 -> S6-> S7 -> S8 -> S1 ->S9

c) Test Generation of DoSelectionSort( )

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Predicate Node | Decision | Possible Outcome | Path | T1 | T2 | T3 | T4 |
| S1 | I <arr.length - 1 | T | P2,P3,P4 |  | X | X | X |
|  |  | F | P1 | X |  |  |  |
| S3 | j < arr.length | T | P3,P4 |  |  | X | X |
|  |  | F | P2 |  | X |  |  |
| S4 | arr[j] < arr[index] | T | P4 |  |  |  | X |
|  |  | F | P3 |  |  | X |  |

|  |  |  |
| --- | --- | --- |
| Test | Input | Output |
| T1 | I=0,j=1,arr.len=3,arr=[3,2,1] | [1,2,3] |
| T2 | I=0,j=1,arr.len=2,arr=[2,1] | [1,2] |
| T3 | I=0,j=1,arr.len=4,arr=[2,1,3] | [1,2,3] |
| T4 | I=0,j=1,arr.len=2,arr=[1,0] | [0,1] |

Question #2: Basis Path Testing (25%)

Cyclomatic complexity and basis-path testing:

Based on your generated program flow graph of QuickSort() Function (given below), please use three different ways to compute its Cyclomatic number.(5%)

Generate a graph matrix based on the given program flow graph of QuickSort () function, and compute the Cyclomatic metric. (5%)

Identify a basis path set (which consists of a number of basis paths) for QuickSort () function. (8%)

List the basis set of test cases (including test inputs and outputs for each test case) (7%)

Ans:

a) Cyclomatic complexity, M(G) of a program flow graph G can be calculated in the following ways:

1. M(G) = Number of regions of G

2. M(G) = |E| - |N| + 2

Where, E is Number of Edges in G

N is number of nodes in G

3. M(G) = |P| + 1

Where, P is number of predicate nodes

So, bases on the above mentioned ways cyclomatic complexity of program graph would be :

1. M(G) = Number of regions of G = **4**

2. The above graph has 11 Edges and 9 Nodes, so

M(G) = |E| - |N| + 2 = 11 - 9 + 2 = **4**

3. The above graph has three predicate nodes, so

M(G) = |P| + 1 = 3 + 1 =**4**

b) Graph matrix based on program flow graph :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 |
| S1 |  | E1 |  |  |  |  |  |  | E11 |
| S2 |  |  | E2 |  |  |  |  |  |  |
| S3 |  |  |  | E3 |  | E7 |  |  |  |
| S4 |  |  | E6 |  | E4 |  |  |  |  |
| S5 |  |  | E5 |  |  |  |  |  |  |
| S6 |  |  |  |  |  |  | E8 |  |  |
| S7 |  |  |  |  |  |  |  | E9 |  |
| S8 | E10 |  |  |  |  |  |  |  |  |
| S9 |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 |  |
| S1 |  | 1 |  |  |  |  |  |  | 1 | 2-1=1 |
| S2 |  |  | 1 |  |  |  |  |  |  | 1-1=0 |
| S3 |  |  |  | 1 |  | 1 |  |  |  | 2-1=1 |
| S4 |  |  | 1 |  | 1 |  |  |  |  | 2-1=1 |
| S5 |  |  | 1 |  |  |  |  |  |  | 1-1=0 |
| S6 |  |  |  |  |  |  | 1 |  |  | 1-1=0 |
| S7 |  |  |  |  |  |  |  | 1 |  | 1-1=0 |
| S8 | 1 |  |  |  |  |  |  |  |  | 1-1=0 |
| S9 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 3+1=**4** |

Hence, **Complexity is 4**

c) There will be total 4 basis path for the above program flow graph :

P1 : S1 -> S9

P2 : S1-> S2 ->S3 ->S6 -> S7 -> S8 -> S1 ->S9

P3: S1-> S2 ->S3 ->S4 -> S3-> S6-> S7 -> S8 -> S1 ->S9

P4 : S1-> S2 ->S3 ->S4 -> S5 -> S3 -> S6-> S7 -> S8 -> S1 ->S9

d) List the basis set of test cases (including test inputs and outputs for each test case)

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Path | Input | Output |
| T1 | P1 | [1] | [1] |
| T2 | P2 | [1,2] | [1,2] |
| T3 | P3 | [2,1] | [1,2] |
| T4 | P4 | [1,2,1] | [1,1,2] |

Question #3: State-Based Software Testing (30%)

Based on Figure 1, please work on the following questions: (Note: S1 is the starting state)

Define two state-based test criteria in details: (5%)

all-state test coverage criteria, and

all-transition test coverage criteria.

Generate a state-based tree based on Figure 1. (10%)

Identify and list all state paths (without redundant paths) from S1 to S4. (10%)

Use a tree-based approach to identify and list a state test set which achieves all-transition- node coverage in Figure 1. (5%)

So

S1

S2

S4

S3

T1

T2

T3

T4

T5

T6

T7

T8

T9

T10

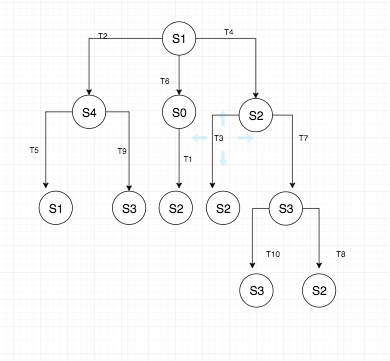
a)

**All-State test coverage criteria**:

Every state is tested at least once using defined test suits. It determines where we cover all the states.

**All-Transition test coverage criteria**: Every transition is tested at least once. It determines whether we cover all the transitions

b)



c) **S1 to S3**

1. S1->T6->S4->T9->S3
2. S1->T4->S2->T7->S3

d) **Test Set:**

Level1:

1. S1->T2->S0
2. S1->T4->S2
3. S1->T6->S4

Level2:

1. S1->T6->S4->T9->S3
2. S1->T4->S2->T3->S2
3. S1->T4->S2->T7->S3
4. S1->T6->S4->T5->S1
5. S1->T2->S0->T1->S2

Level3:

1. S1->T4->S2->T7->S3->T8->S2
2. S1->T4->S2->T7->S3->T10->S3

Question #4: AI System Testing and Tools (20%)

Please search the web to identify and discover your answer to the following questions.

List and capture 5 research papers relating AI system testing. (5%)

|  |  |
| --- | --- |
| Paper Number | URL |
| **INSPEC Accession Number: 4239523**  **DOI**: [10.1109/DASC.1991.177202](https://doi.org/10.1109/DASC.1991.177202) | https://ieeexplore.ieee.org/document/177202 |
| **INSPEC Accession Number:**8748442  **DOI:**[10.1109/QSIC.2005.12](https://doi.org/10.1109/QSIC.2005.12) | http://ieeexplore.ieee.org/abstract/document/1579143/ |
| **INSPEC Accession Number: 5513649**  **DOI**: [10.1109/AUTEST.1996.547718](https://doi.org/10.1109/AUTEST.1996.547718) | https://ieeexplore.ieee.org/document/547718 |
| **INSPEC Accession Number:**5870123  **DOI:**[10.1109/TEST.1997.639719](https://doi.org/10.1109/TEST.1997.639719) | http://ieeexplore.ieee.org/abstract/document/639719/ |
| **INSPEC Accession Number: 14684783**  DOI: [**10.1109/ICCSE.2014.6926420**](https://doi.org/10.1109/ICCSE.2014.6926420) | https://ieeexplore.ieee.org/document/6926420 |

List and explain top 5 AI testing tools in a list with detailed tool profile information. (5%)

Compare them and present the following table. (10%)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tool name  URL | Maker | Major Features | Technology | Focus Area | Platform | Limitations |
| Testim  URL:  https://www.testim.io | Testim | Group several steps with a click. Reuse that group in several tests.,Pass parameters to groups and tests.,Use variables,Do Data Driven Testing | NA | NA | NA | NA |
| https://www.appvance.ai | appvance | Significantly less bugs slip into generation with Appvance IQ since its tremendous and extensive test creation and execution abilities find generally surrenders. | NA | NA | NA | NA |
| test.ai  URL: http://test.ai/ | Test.ai | Easy to setup, no programming required and its pretty low maintenance tool.  It is helpful with regards to getting the performance metrics. | Viewport Meta, IPhone / Mobile Compatible, and Google Analytics. | NA | NA | NA |
| [Functionize](https://www.functionize.com/)  URL: https://www.functionize.com/ | [Functionize](https://www.functionize.com/) | Scripting not required for writing test, execute numerous tests in minutes, Complete in-depth analyses. | Works for mobile testing, visual testing, cross browser testing | NA | NA | NA |
| Mabl  URL:  https://www.mabl.com/ | Mabl | No scripting required for creating test cases, Auto healing tests, Machine driven regression testing | Integrates with tools: JIRA, JenkinsCircleci | NA | NA | NA |