**Part 1 Example:**

The schedule manager schedules a meeting to (a) time slot(s) as part of the 5-day schedule for the event. To accomplish this, the user must input three items:

1. The day (1, 2, 3, 4, or 5), chosen from a drop-down list.

2. The time slot(s), chosen through some type of user interface. Each day of the event lasts from 10am-6pm. There are 8, 1-hour time slots available per day (10-11am, 11am-12pm, 12-1pm, 1-2pm, 2-3pm, 3-4pm, 4-5pm, 5-6pm). A meeting can be scheduled for a 1-hour, 2-hour, or 3-hour time slot.

3. The name of the meeting (chosen from a drop-down list of meetings that the system already knows about)

The output would be a meeting being scheduled for a particular date and time (as reflected in the schedule). For example, if one of my test cases consisted of: {Day 3, 11am-1pm, The Go-Go's}, the expected output for this test case would be the Go-Go's being scheduled for Day3 of the festival in the 11am-1pm time slot.

**(The minimum time unit is 1 hour. Minutes are not considered in this question)**

Please provide the **Equivalence Partitioning** and **Boundary Value Analysis** of this use case and provide relevant **Test Cases Design**.

Equivalence classes

|  |  |  |
| --- | --- | --- |
| Parameter name | equivalence classes | No. |
| day | 1,2,3,4,5 | D1 |
| null | D2 |
| time | 1 | T1 |
| 2 | T2 |
| 3 | T3 |
| [4,8] | T4 |
| null | T5 |
| name | Has name | N1 |
| null | N2 |

Boundary value analysis

Day: 1, 5, null

Time: 10-11am, 5-6pm, 10-12am, 4-6pm, 10-1am, 3-6pm, 10-14am, 2-6pm, null

Name: has name, null

Test Case Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case No. | Input Value 1 | Input Value 2 | Input Value 3 | Expected Output | Description |
| 1 | 1 | 10-11am | name |  | D1T1N1 |
| 2 | 1 | 5-6pm | name |  | D1T1N1 |
| 3 | 5 | 10-11am | name |  | D1T1N1 |
| 4 | 5 | 5-6pm | name |  | D1T1N1 |
| 5 | 1 | 10-12am | name |  | D1T2N1 |
| 6 | 1 | 4-6pm | name |  | D1T2N1 |
| 7 | 5 | 10-12am | name |  | D1T2N1 |
| 8 | 5 | 4-6pm | name |  | D1T2N1 |
| 9 | 1 | 10-1am | name |  | D1T3N1 |
| 10 | 1 | 3-6pm | name |  | D1T3N1 |
| 11 | 5 | 10-1am | name |  | D1T3N1 |
| 12 | 5 | 3-6pm | name |  | D1T3N1 |
| 13 | null | 10-11am | name |  | D2T1N1 |
| 14 | 1 | 10-14am | name |  | D1T4N1 |
| 15 | 1 | null | name |  | D1T5N1 |
| 16 | 1 | 10-11am | null |  | D1T1N2 |

**Part 2 Example:**

Now we have two code written by some programmers which are shown below, please provide the Control Flow Testing process of each code, including **Control Flow Graph, Cyclomatic complexity and Basis Set.** **And you can report the bugs if you find any.**

Code 1:

1. function radixSort(arr, maxDigit) {

2. varmod = 10;

3. vardev = 1;

4. for(vari = 0; i < maxDigit; i++, dev \*= 10, mod \*= 10) {

5. for(varj = 0; j < arr.length; j++) {

6. varbucket = parseInt((arr[j] % mod) / dev);

7. if(counter[bucket]==null) {

8. counter[bucket] = [];

9. }

10. counter[bucket].push(arr[j]);

11. }

12. varpos = 0;

13. for(varj = 0; j < counter.length; j++) {

14. varvalue =null;

15. if(counter[j]!=null) {

16. while((value = counter[j].shift()) !=null) {

17. arr[pos++] = value;

18. }

19. }

20. }

21. }

22. return arr;

23. }

Code 2:

1 void Merge(int a[], int left, int mid, int right)

2 {

3 int len = right - left + 1;

4 int \*temp = new int[len];

5 int k = 0;

6 int j = mid + 1;

7 while (i <= mid && j <= right)

8 {

9 temp[k++] = a[i] <= a[j] ? a[i++] : a[j++];

10 }

11 while (i <= mid)

12 {

13 temp[k++] = a[i++];

14 }

15 while (j <= right)

16 {

17 temp[k++] = a[j++];

18 }

19 for (int k = 0; k < len; k++)

20 {

21 a[left++] = temp[k];

22 }

23 }