INTRODUCTION

Testing is an expensive part of software development process often consisting of approximately 50% of overall budget. It also fails to find many of the problems in software. Testing is thus a difficult and expensive process and the development of efficient, effective test technique is the major research paper. Developing effective and efficient techniques has been a major problem when creating test cases; this has been the point of discussion for many years. There are several well known techniques associated with creating test cases for a system. There are different testing techniques which are applied in different phases of testing process e.g. black box and white box testing. Boundary Value Analysis is one of the most popular Black Box testing techniques.

Boundary Value Analysis[7] means an input value may be on the boundary, just below the boundary (upper side), just above the boundary (lower side). It is a test selection technique that targets fault in application at the boundaries of equivalent class.

A great number of errors occur at the boundaries of Input domain rather than in the “center”. It is for this reason that BVA has been developed as testing technique. S/w is very binary[6]—something is either true or false. If an operation is performed on a range of numbers, odds majority of the numbers in the middle but may be made a mistake at the edge.

In case of ranges for boundary value analysis it is useful to select the boundary element of range and invalid values just beyond the two ends. So, If the range is

\[ 0.0 \leq x \leq 1.0, \]

the test cases are (0.0, 1.0) for valid input and (-0.1, 1.1) for invalid inputs. Similarly if the input is a list attention should be focused on the first and last elements of the list.

When you are presented with the software test problem that involves identifying boundaries, look for the following types

1. numeric
2. position
3. quantity

In addition to identifying boundaries using equivalence classes, it is also possible and recommended that boundaries be identified based on the selection among input variable. Once Input domain has been identified, test selection using
Boundary Value Analysis proceeds as follow:

1. Partition Input domain using uni-dimensional partitioning, this leaves to as many partitions as there are input variables. Alternatively, a single partition of an input domain can be created using multidimensional partitioning.

2. Identifying the boundary for each partition. Boundaries may also be identified using special relationship among the inputs.

3. Select test data such that each boundary value occurs in at least one test input.

Test Case Generation for Numerical variable in Boundary Value Analysis:

Consider a simple program7 to classify a triangle. Its input is a triple of positive integers (x, y and z) and the data type for input parameters ensures that these will be integers greater than 0 and less than or equal to 100. The program may be one of the following words: [Scalene; Isosceles; Equilateral; Not a Triangle]

Standard Boundary Value Analysis test cases

\[ \text{min} = 1, \text{min}+ = 2, \text{nom} = 100, \text{max-} = 199, \text{max} = 200 \]

<table>
<thead>
<tr>
<th>Case</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Expected output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>Equilateral</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>2</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>Isosceles</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
<td>199</td>
<td>Isosceles</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
<td>200</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>2</td>
<td>100</td>
<td>Isosceles</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>199</td>
<td>100</td>
<td>Isosceles</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>200</td>
<td>100</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>1</td>
<td>100</td>
<td>Isosceles</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>100</td>
<td>100</td>
<td>Isosceles</td>
</tr>
<tr>
<td>11</td>
<td>199</td>
<td>100</td>
<td>100</td>
<td>Isosceles</td>
</tr>
<tr>
<td>12</td>
<td>200</td>
<td>100</td>
<td>100</td>
<td>Not a triangle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Expected output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Equilateral</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>199</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>200</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>Isosceles</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>100</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>199</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>200</td>
<td>Not a triangle</td>
</tr>
</tbody>
</table>

Worst case test cases (60-125)

<table>
<thead>
<tr>
<th>Case</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Expected output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Equilateral</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>Isosceles</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>199</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>200</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>Isosceles</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>100</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>199</td>
<td>Not a triangle</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>200</td>
<td>Not a triangle</td>
</tr>
</tbody>
</table>

Limitations of Boundary Value Analysis

Boundary Value Analysis1 works well when the Program Under Test (PUT) is a “function of

several independent variables that represent bounded physical quantities*. When these conditions are met, BVA works well but when they are not, we can find deficiencies in the results.

The reason for this poor performance is that BVA cannot compensate or take into consideration the nature of a function or the dependencies between its variables. This lack of intuition or understanding for the variable nature means that BVA can be seen as quite rudimentary.

**Test Case for Non-Numerical Variable: Strings**

There are several approaches to Boundary Value Analysis, based on arguments in which we choose a set of Test inputs for a boundary B such that if there is a boundary shift in B with in the implementation, then it is likely that at least one value from T will be in the wrong sub-domain in the implementation.

In order to simplify, the Normal Boundary conditions are the ones defined in specification or evident when using the software. Some boundaries that are internal to the software are not necessarily apparent to an end-user but still needs to be check by software tester. These are known as sub-boundary or Internal Boundary conditions.

The most common sub-boundary condition is ASCII Character table:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define size 3

void main()
{
    char strsrc[size];
    char strtmp[size];
    clscr();
    printf("Enter String:=");
    gets(strsrc);
    strcpy(strtmp,strupr(strsrc));
    strrev(strtmp);
    if(strcmp(strsrc,strtmp)==0)
        printf("Enter string %s is palindrome",strsrc);
    else
        printf("Entered string %s is not palindrome",strsrc);
    getch();
}
```

We will demonstrate through an example to check whether the string is palindrome or not:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define size 3

void main()
{
    char strsrc[size];
    char strtmp[size];
    clscr();
    printf("Enter String:=");
    gets(strsrc);
    strcpy(strtmp,strupr(strsrc));
    strrev(strtmp);
    if(strcmp(strsrc,strtmp)==0)
        printf("Enter string is palindrome",strsrc);
    else
        printf("Entered string is not palindrome",strsrc);
    getch();
}
```

**Test case Generation**

Input Domain [A-Z]

The boundary value test cases are:

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N</td>
<td>U</td>
<td>L</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>O</td>
<td>H</td>
<td>DC1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>S T</td>
<td>X</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>E T</td>
<td>X</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>E C T</td>
<td>DC4</td>
<td>$</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>E N G</td>
<td>N A K</td>
<td>%</td>
<td>5</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>A C K</td>
<td>S Y N</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>B E L</td>
<td>E T B</td>
<td>'</td>
<td>7</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>B S</td>
<td>C A N</td>
<td>(</td>
<td>8</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>H T</td>
<td>E M</td>
<td>)</td>
<td>9</td>
<td>I</td>
</tr>
<tr>
<td>A</td>
<td>L F</td>
<td>S U L E</td>
<td>*</td>
<td>J</td>
<td>Z</td>
</tr>
<tr>
<td>B</td>
<td>V T</td>
<td>E S C</td>
<td>+</td>
<td>;</td>
<td>K</td>
</tr>
<tr>
<td>C</td>
<td>F F</td>
<td>F S</td>
<td>,</td>
<td>&lt;</td>
<td>L</td>
</tr>
<tr>
<td>D</td>
<td>C R</td>
<td>G S</td>
<td>-</td>
<td>=</td>
<td>M</td>
</tr>
<tr>
<td>E</td>
<td>S O</td>
<td>R S</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
</tr>
<tr>
<td>F</td>
<td>S</td>
<td>U S</td>
<td>/</td>
<td>?</td>
<td>O</td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>Test</th>
<th>Alphabet1</th>
<th>Alphabet2</th>
<th>Alphabet3</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>4</td>
<td>Y</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>5</td>
<td>Z</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>B</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>8</td>
<td>M</td>
<td>Y</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>Z</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>10</td>
<td>M</td>
<td>M</td>
<td>A</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>M</td>
<td>B</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>M</td>
<td>Y</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>13</td>
<td>M</td>
<td>M</td>
<td>A</td>
<td>Not palindrome</td>
</tr>
</tbody>
</table>

Robustness testing
There are 4 additional test cases which are outside the legitimate Input domain. In addition to the aforementioned 5 testing values (min, min+, nom, max-, max) we use two more values for each variable (min-, max+), which are designed to fall just outside of the input range.

<table>
<thead>
<tr>
<th>Test</th>
<th>Alphabet1</th>
<th>Alphabet2</th>
<th>Alphabet3</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>@</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>4</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>5</td>
<td>Y</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>6</td>
<td>Z</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>8</td>
<td>M</td>
<td>@</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>10</td>
<td>M</td>
<td>B</td>
<td>A</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>11</td>
<td>M</td>
<td>Y</td>
<td>B</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>Z</td>
<td>Y</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>13</td>
<td>M</td>
<td>L</td>
<td>A</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>14</td>
<td>M</td>
<td>M</td>
<td>@</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>15</td>
<td>M</td>
<td>M</td>
<td>A</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>16</td>
<td>M</td>
<td>M</td>
<td>B</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>17</td>
<td>M</td>
<td>M</td>
<td>Y</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>18</td>
<td>M</td>
<td>M</td>
<td>Z</td>
<td>Not palindrome</td>
</tr>
<tr>
<td>19</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>Not palindrome</td>
</tr>
</tbody>
</table>

Hence total test cases in Robustness testing are 6n+1, where n is number of input variable.i.e 6*3+1=19 cases.

Worst-Case Testing
If we reject single fault assumption theory of reliability and may like to see what happens when more than one variable has an extreme value. It is called Worst-Case Analysis.
It is more thorough in the sense that boundary value test cases are a proper subset of worst case test cases. It requires more effort. Worst case testing for a function of n variable generates $5^n$ test cases as opposed to $4n+1$ test case for Boundary value Analysis. E.g. $5^3=125$ test cases given in the following table.

<table>
<thead>
<tr>
<th>Test</th>
<th>Alphabet 1</th>
<th>Alphabet 2</th>
<th>Alphabet 3</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>A</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>A</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>A</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>B</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>B</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>B</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>M</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>12</td>
<td>A</td>
<td>M</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>13</td>
<td>A</td>
<td>M</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>14</td>
<td>A</td>
<td>M</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>15</td>
<td>A</td>
<td>M</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>16</td>
<td>A</td>
<td>Y</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>17</td>
<td>A</td>
<td>Y</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>18</td>
<td>A</td>
<td>Y</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>19</td>
<td>A</td>
<td>Y</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>20</td>
<td>A</td>
<td>Y</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>21</td>
<td>A</td>
<td>Z</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>22</td>
<td>A</td>
<td>Z</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>23</td>
<td>A</td>
<td>Z</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>24</td>
<td>A</td>
<td>Z</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>25</td>
<td>A</td>
<td>Z</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>26</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>27</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>28</td>
<td>B</td>
<td>A</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>29</td>
<td>B</td>
<td>A</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>30</td>
<td>B</td>
<td>A</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>31</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>32</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>33</td>
<td>B</td>
<td>B</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>34</td>
<td>B</td>
<td>B</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>35</td>
<td>B</td>
<td>B</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>36</td>
<td>B</td>
<td>M</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>37</td>
<td>B</td>
<td>M</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>38</td>
<td>B</td>
<td>M</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>39</td>
<td>B</td>
<td>M</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>40</td>
<td>B</td>
<td>M</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>41</td>
<td>B</td>
<td>Y</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>42</td>
<td>B</td>
<td>Y</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>43</td>
<td>B</td>
<td>Y</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>B</td>
<td>Y</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>45</td>
<td>B</td>
<td>Y</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>46</td>
<td>B</td>
<td>Z</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>47</td>
<td>B</td>
<td>Z</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>48</td>
<td>B</td>
<td>Z</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>49</td>
<td>B</td>
<td>Z</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>50</td>
<td>B</td>
<td>Z</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>51</td>
<td>M</td>
<td>A</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>52</td>
<td>M</td>
<td>A</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>53</td>
<td>M</td>
<td>A</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>54</td>
<td>M</td>
<td>A</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>55</td>
<td>M</td>
<td>A</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>56</td>
<td>M</td>
<td>B</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>57</td>
<td>M</td>
<td>B</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>58</td>
<td>M</td>
<td>B</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>59</td>
<td>M</td>
<td>B</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>60</td>
<td>M</td>
<td>B</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>61</td>
<td>M</td>
<td>M</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>62</td>
<td>M</td>
<td>M</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>63</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>64</td>
<td>M</td>
<td>M</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>65</td>
<td>M</td>
<td>M</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>66</td>
<td>M</td>
<td>Y</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>67</td>
<td>M</td>
<td>Y</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>68</td>
<td>M</td>
<td>Y</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>69</td>
<td>M</td>
<td>Z</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>70</td>
<td>M</td>
<td>Z</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>71</td>
<td>M</td>
<td>Z</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>72</td>
<td>M</td>
<td>Z</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>73</td>
<td>M</td>
<td>Z</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>74</td>
<td>M</td>
<td>Z</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>75</td>
<td>M</td>
<td>Z</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>76</td>
<td>Y</td>
<td>A</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>77</td>
<td>Y</td>
<td>A</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>78</td>
<td>Y</td>
<td>A</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>79</td>
<td>Y</td>
<td>A</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>80</td>
<td>Y</td>
<td>A</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>81</td>
<td>Y</td>
<td>B</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>82</td>
<td>Y</td>
<td>B</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>83</td>
<td>Y</td>
<td>B</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>84</td>
<td>Y</td>
<td>B</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>85</td>
<td>Y</td>
<td>B</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>86</td>
<td>Y</td>
<td>M</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>87</td>
<td>Y</td>
<td>M</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>88</td>
<td>Y</td>
<td>M</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>89</td>
<td>Y</td>
<td>M</td>
<td>Z</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>90</td>
<td>Y</td>
<td>M</td>
<td>Y</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>91</td>
<td>Y</td>
<td>Y</td>
<td>A</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>92</td>
<td>Y</td>
<td>Y</td>
<td>B</td>
<td>Not Palindrome</td>
</tr>
<tr>
<td>93</td>
<td>Y</td>
<td>Y</td>
<td>M</td>
<td>Not Palindrome</td>
</tr>
</tbody>
</table>
Future Recommendation

Although ASCII is still popular as the common means for software to represent character data it is being replaced by a new standard called Unicode. ASCII using only 8 bit, can represent only 256 different characters. Unicode which uses 16 bits can represent 65536 characters.

**CONCLUSION**

This article has investigated the problem of generating test cases for non numerical values. We have taken ASCII characters for generation of robustness, and worst case testing.

1. Boundary value test case-4n+1
2. Robustness testing -6n+1
3. Worst case testing -5n

B.V.A works well for the program with independent Input value where input value Should be truly independent This does not make sense for Boolean variables where extreme value are True and False but no clear choice is available for others like nominal, just above boundary and just below boundary.

Since we have already proved that BVA is suitable for numerical values, where the range can be determined. In this article we have also proved that it can also be applied for non-numerical values where the range cannot be specific, we also use special characters as Inputs.
REFERENCES

5. Gregor Snelting, Torsten Robschink, Jens Krinke, Software Engineering and Methodology, December 06, Volume 15
6. Ron Patton, Software Testing, SAMS Techmedia
7. KK Aggarwal, Yogesh Singh, Software Engineering, New Age International Publishers