HMR

Institute of Technology and Management

Laboratory File

SOFTWARE TESTING & QUALITY ASSURANCE

For 4th Year Student

Department : Computer Science & Engineering

ETCS – 453

Submitted To :	**Submitted By :**
Ms. Namrata Sukhija	**Mohd. Aftab Alam**
CSE Department	**CSE – 7A – 02013302717**

Mohd. Aftab Alam
02013302717

INDEX

24/08/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 1

Aim :

Introduction to Software Testing and Quality Assurance.

Explain the following terms in brief :

1. Software

Software is a collection of data, instructions or program that tell the computer how to work using the hardware attached with it. Without software, the hardware is just a useless piece of machine. There are two types of software : system software and application software.

Example : Visual Studio Code – it is a free source code editor made by Microsoft for Windows, Linux and macOS. It is the most popular developer environment tool.

2. Program

Program is a set of instructions given by the user to a computer to perform a specific task. It can be written in any programming language like C/C++, Python, Java, Swift, JavaScript, etc. A program in execution state is called process.

Example : Python program to print “Manchester United” is shown below –

```
print('Manchester United')
```

3. Software Engineering

Software Engineering is a discipline whose aim is the production of quality software, software that is delivered on time, within budget and that satisfies its requirements. The result of software engineering is an efficient and reliable software product.

4. Software Testing

Software Testing is a process to check whether the actual results match the expected results and to ensure that the software system is defect free. It is important because software bugs could be expensive or even dangerous.

5. Software Testing Process / V Model of Testing

This process involves 5 steps which are given below –

- Planning and Control

Planning – it involves producing a document that describes an overall approach and test objectives. It involves reviewing the test basis, identifying the test conditions based on analysis of test items, writing test cases and designing the test environment.

Control – this is the activity of comparing actual progress against the plan, and reporting the status, including deviations from the plan. It involves taking actions necessary to meet the mission and objectives of the project.

- Analysis and Design – major tasks : to review the test basis, to identify test conditions, to design the tests, to design the test environment set-up and identify the required infrastructure and tools

- Implementation and Execution

Implementation – major tasks : to develop and prioritize test cases by using techniques and create test data for those tests, to create test suites from the test cases for efficient test execution, to re-execute the tests that previously failed in order to confirm a fix, to log the outcome of the test execution and to compare actual results with expected results.

Execution – it involves actually running the specified test on a computer system either manually or by using an automated test tool. It is a fundamental test process in which actual work is done.

- Evaluating Exit Criteria and Reporting – it is a process defining when to stop testing. It depends on coverage of code, functionality or risk. Basically it also depends on business risk, cost and time and vary from project to project. Major tasks : to assess if more tests are needed or if the exit criteria specified should be changed and to write a test summary report for stakeholders.

- Test Closure Activities – these activities are done when software is ready to be delivered. Major tasks : to check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved, to finalize and archive test ware such as scripts, test environments, etc. for later reuse, to handover the test ware to the maintenance organization and to evaluate how the testing went and learn lessons for future releases and projects.

6. Why should we do Testing ?

Software Testing helps to identify errors, gaps and missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss.

7. What should we Test ?

We should test a program for all possible valid and invalid inputs or we can say all possible execution paths. For this we require effective planning, strategies and sufficient resources. We should also test the program for very large numbers, very small numbers, numbers that are close to each other, negative numbers, some extreme cases, characters, and some strange cases.

8. Who should do the Testing ?

For a small software, the developer should be responsible for testing and there are various plus points of testing done by the developer. Apart from this, there are professionals who work as full time software testers. There are various companies who hire these professionals to test their products and services.

9. Quality Assurance

Quality Assurance (QA) is a way of preventing mistakes and defects in manufactured products and avoiding problems when delivering products or services to customers. It comprises administrative and procedural activities implemented in a quality system so that requirements and goals for a product, service or activity will be fulfilled.

10. Quality Control

Quality Control (QC) is a process by which entities review the quality of all factors involved in production. Inspection is a major component of quality control, where physical product is examined visually (or the end results of a service are analyzed).

11. Verification and Validation

Verification – the process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. It is a static practice of verifying documents, design, code and program. It will help to determine whether the software is of high quality, but it will not ensure that the system is useful. It is concerned with whether the system is well-engineered and error-free.

Validation – the process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. It is the process of evaluating the final product to check whether the software meets the customer expectations and requirements. It is a dynamic mechanism of validating and testing the actual product.

12. Error, Defect, Bug and Failure

Error – it is a human mistake. It appears not only due to the logical mistake in the code made by the developer. Anyone in the team can make mistakes during the different phases of software development.

Defect – it is a variance between expected and actual results. An error that the tester finds is known as defect. A defect in a software product reflects its inability or inefficiency to comply with the specified requirements and criteria and, subsequently, prevent the software application from performing the desired and expected work. It is also known as ***Fault.***

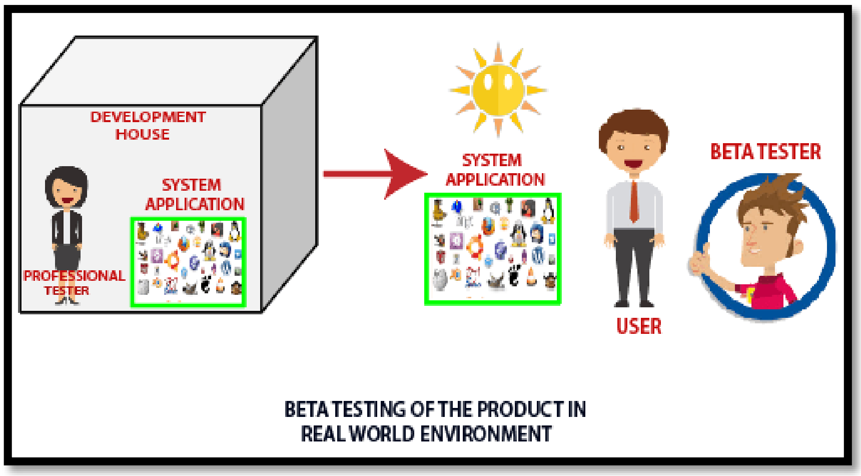
Bug – it occurs because of some coding error and leads a program to malfunction. It may also lead to a functional issue in the product. These are fatal errors that could block a functionality, results in a crash, or cause performance bottlenecks.

Failure – it is a consequence of a defect. It is the observable incorrect behavior of the system. It occurs when the software fails to perform in the real environment. Not all defects result in failures; some remain inactive in the code, and we may never notice them.

13. Alpha, Beta and Acceptance Testing

Alpha Testing – it is performed to identify all possible issues and bugs before releasing the final product to the end users. It is carried out by the testers who are internal employees of the organization. The main goal is to identify the tasks that a typical user might perform and test them.

Beta Testing – it is performed by "real users" of the software application in "real environment". It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of beta testing. This testing helps to test products in customer's environment.



Acceptance Testing – it is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of black box testing where the number of required users involved test the acceptance level of the system.

14. Deliverables and Milestones

Deliverables – it is a distinct, tangible or intangible outcome of your project that is produced during the project's course. These are developed by the project team members in alignment with the overall objectives of the project.

Milestones – these are checkpoints in the project that help you chart progress throughout the course of the project. These control points help identify that a number of tasks or key deliverables have been completed allowing you to move on to the next phase of your project.

15. Static and Dynamic Testing

Static Testing – testing which checks the application without executing the code. It is a verification process. It is performed in the white box testing phase, where the programmer checks every line of the code before handling over to the test engineer.

Dynamic Testing – testing which is done when the code is executed at the run time environment. It is a validation process where functional and non-functional testing is performed. It is performed to check whether the application or software is working fine during and after the installation.

16. White Box and Black Box Testing

White Box Testing – here the internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to software testers.

White Box Testing Techniques :

- Data Flow Testing
- Control Flow Testing
- Branch Testing
- Statement Testing
- Decision Testing

Black Box Testing – here the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. It mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as ***Behavioral Testing.***

Black Box Testing Techniques :

- Decision Table Technique
- Boundary Value Technique
- State Transition Technique
- All-Pair Testing Technique
- Cause-Effect Technique
- Equivalence Partitioning Technique
- Error Guessing Technique
- Use Case Technique

17. Why 100% Testing is not possible ?

100% Testing is not possible because of the following reasons :

- The domain of possible inputs of a program is too large to be completely used in testing a system. There are both valid inputs and invalid inputs.
- The design issues may be too complex to completely test. The design may have included implicit design decisions and assumptions.
- It may not be possible to create all possible execution environments of the system.

18. Limitations of Testing

Limitations of Testing are given below :

- Testing can be used to show the presence of errors, but never to show their absence. It can only identify the known issues or errors. It gives no idea about defects still uncovered. Testing cannot guarantee that the system under test is error free.
- Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.
- Software testing does not help in finding root causes which resulted in injection of defects in the first place. Locating root causes of failures can help us in preventing injection of such faults in future.

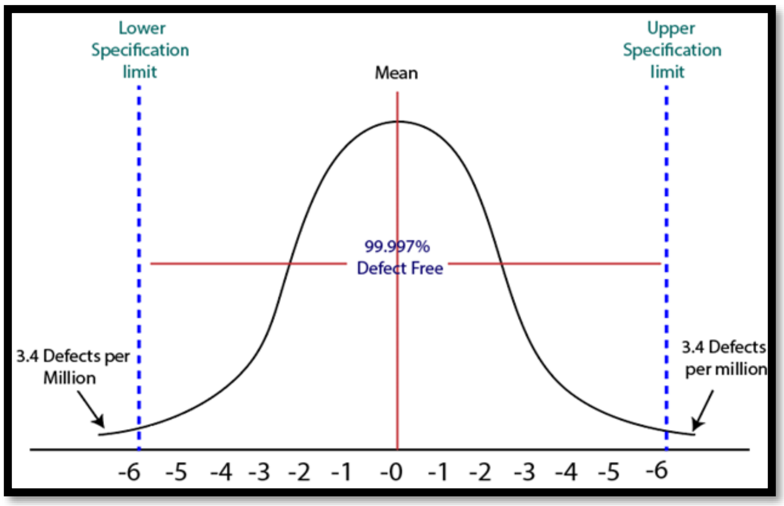
19. Quality and Reliability

Quality – it is defined in terms of its fitness of purpose; that is, a quality product does precisely what the users want it to do. Characteristics of quality are – portability, usability, reusability, correctness and maintainability.

Reliability – it means operational reliability. It is described as the ability of a system or component to perform its required functions under static conditions for a specific period. It is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

20. Six Sigma Concepts

Six Sigma is the process of improving the quality of the output by identifying and eliminating the cause of defects and reduce variability in manufacturing and business processes. A six sigma method is one in which ***99.997%*** of all the opportunities to produce some features of a component are statistically expected to be free of defects.

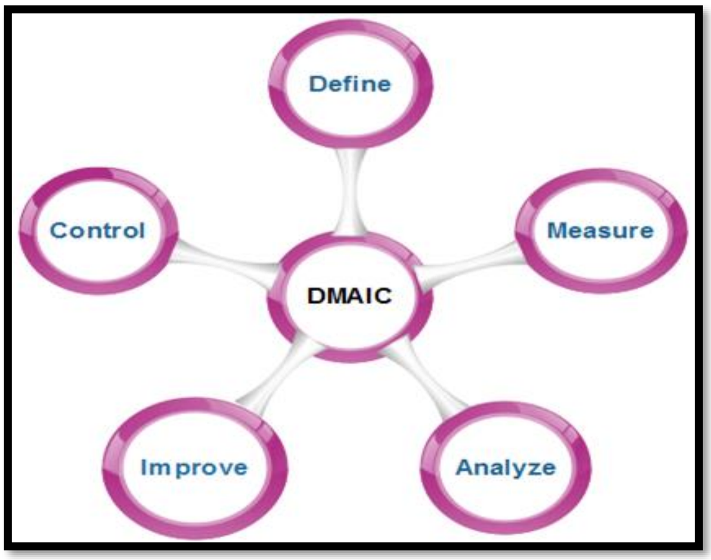


Characteristics of Six Sigma :

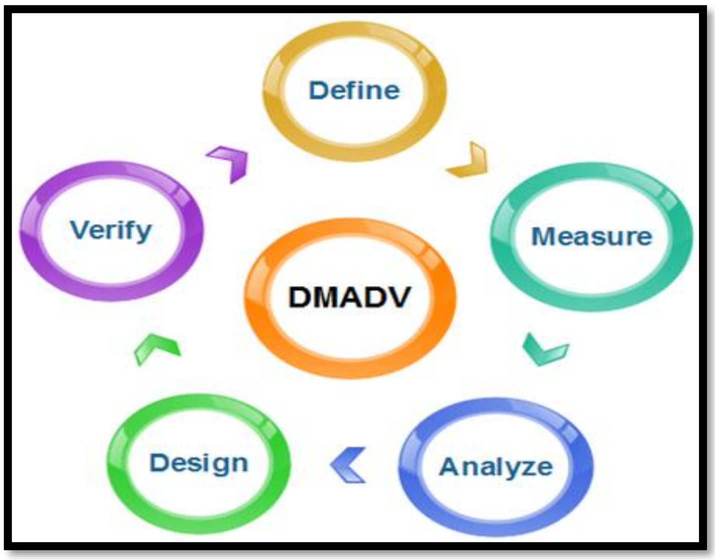
- Statistical Quality Control
- Methodical Approach
- Fact and Data-Based Approach
- Project and Objective-Based Focus
- Customer Focus
- Teamwork Approach to Quality Management

Six Sigma Methodologies :

DMAIC - It specifies a data-driven quality strategy for improving processes. This methodology is used to enhance an existing business process.



DMADV - It specifies a data-driven quality strategy for designing products and processes. This method is used to create new product designs or process designs in such a way that it results in a more predictable, mature, and detect free performance.



31/08/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 2

Aim :

(a) To determine the nature of roots of a Quadratic Equation, its input is triple of positive integers (say a, b, c) and values may be from interval [1, 100]. The output may have one of the following :

Real & Distinct Roots, Imaginary Roots or Real & Equal Roots.

Design the Boundary Value Test Cases.

Algorithm :

- Take 3 inputs from the user for the quadratic equation.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- Else if condition is true, check the nature of the roots.
- If the Discriminant is greater than 0, Real & Distinct Roots.
- If the Discriminant is less than 0, Imaginary Roots.
- If the Discriminant is equal to 0, Real & Equal Roots.
- According to the formula ***4n+1,*** there will be 13 test cases, where n is number of inputs.

Code :

```
#include <iostream>
#include <math.h>

using namespace std;

int bva(int, int, int);

int main()
{
    int min, max;
    int x, y, z;
    cout << "Enter Range : ";
    cin >> min >> max;
    if (min < 0 || max > 100)
    {
        cout << "Invalid Range";
        return 0;
    }
    int nominal = (min + max) / 2;
    int values[] = {min, min + 1, nominal, max - 1, max};
    cout << "a\tb\tc\tOutput\t\t\tRoots" << endl;
    for (int i = 0; i < 5; i++)
    {
        bva(values[i], nominal, nominal);
    }
    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, values[i], nominal);
    }
    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, nominal, values[i]);
    }

    cout << "Enter the Coefficients (a, b, c) : ";
    cin >> x >> y >> z;
    cout << "a\tb\tc\tOutput\t\t\tRoots" << endl;
    bva(x, y, z);
    return 0;
}

int bva(int a, int b, int c)
{
    if (a == 0)
    {
        cout << "Not a Quadratic Equation\n";
        return 0;
    }
    int d = b * b - 4 * a * c;
    double sqrt_val = sqrt(abs(d));
    cout << a << "\t" << b << "\t" << c << "\t";
    if (d < 0)
    {
        cout << "Imaginary Roots\t\t";
        cout << -(double)b / (2 * a) << "+i" << sqrt_val << ", ";
        cout << -(double)b / (2 * a) << "-i" << sqrt_val << endl;
    }
    else if (d == 0)
    {
        cout << "Real and Equal\t\t";
        cout << -(double)b / (2 * a) << endl;
    }
    else
    {
        cout << "Real and Distinct\t";
        cout << (double)(-b + sqrt_val) / (2 * a);
        cout << ", " << (double)(-b - sqrt_val) / (2 * a) << endl;
    }
    return 0;
}
```

<u>**31/08/2020**</u> <u>**Mohd. Aftab Alam**</u>
<u>**02013302717**</u>

<u>Boundary Value Analysis</u> :

Range : R [1, 100]

Domain : Minimum = 1

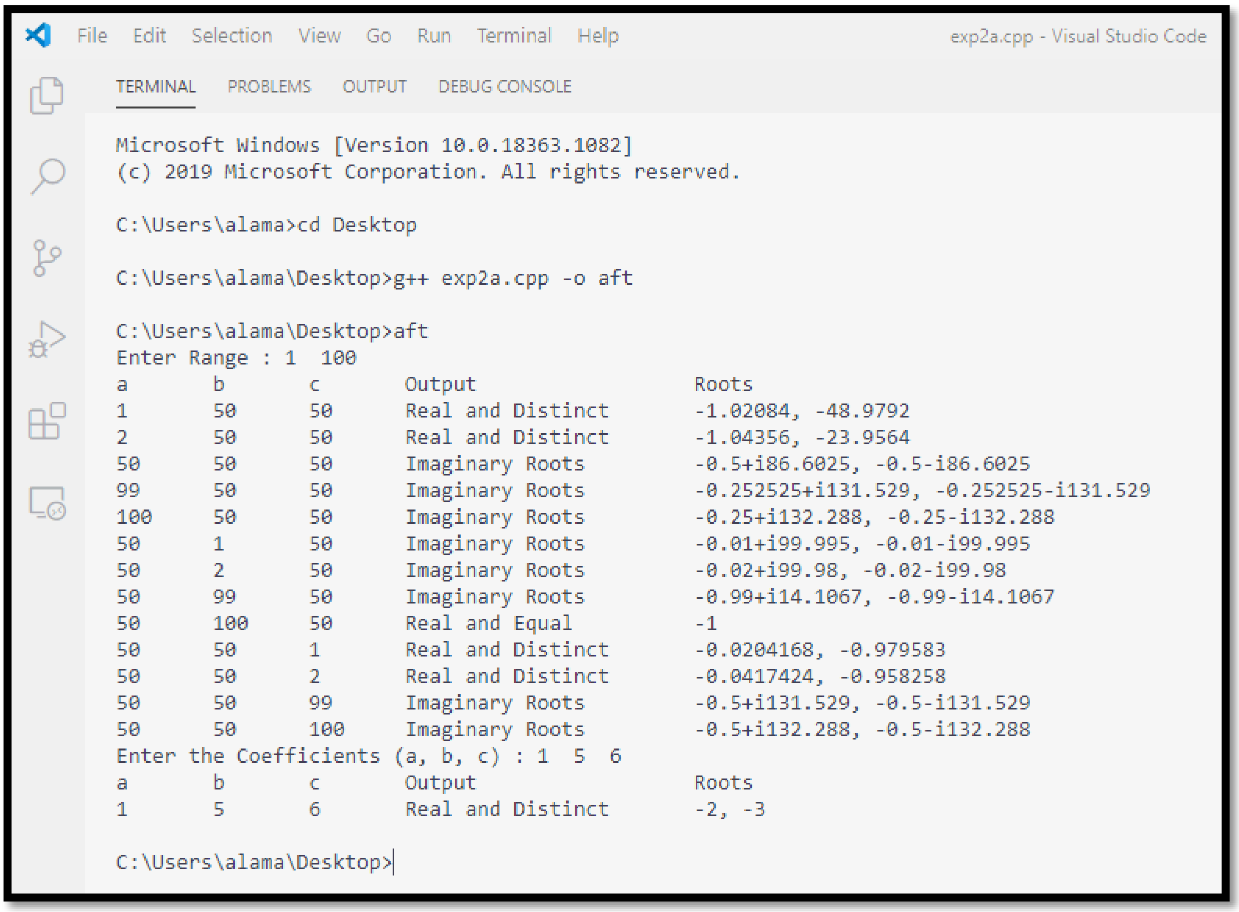
Above Minimum = 2

Nominal = 50

Below Maximum = 99

Maximum = 100

<u>Output</u> :


File Edit Selection View Go Run Terminal Help
exp2a.cpp - Visual Studio Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Desktop

C:\Users\alama\Desktop>g++ exp2a.cpp -o aft

C:\Users\alama\Desktop>aft
Enter Range : 1  100
a       b       c       Output                  Roots
1       50      50      Real and Distinct       -1.02084, -48.9792
2       50      50      Real and Distinct       -1.04356, -23.9564
50      50      50      Imaginary Roots         -0.5+i86.6025, -0.5-i86.6025
99      50      50      Imaginary Roots         -0.252525+i131.529, -0.252525-i131.529
100     50      50      Imaginary Roots         -0.25+i132.288, -0.25-i132.288
50      1       50      Imaginary Roots         -0.01+i99.995, -0.01-i99.995
50      2       50      Imaginary Roots         -0.02+i99.98, -0.02-i99.98
50      99      50      Imaginary Roots         -0.99+i14.1067, -0.99-i14.1067
50      100     50      Real and Equal          -1
50      50      1       Real and Distinct       -0.0204168, -0.979583
50      50      2       Real and Distinct       -0.0417424, -0.958258
50      50      99      Imaginary Roots         -0.5+i131.529, -0.5-i131.529
50      50      100     Imaginary Roots         -0.5+i132.288, -0.5-i132.288
Enter the Coefficients (a, b, c) : 1  5  6
a       b       c       Output                  Roots
1       5       6       Real and Distinct       -2, -3

C:\Users\alama\Desktop>
```

Aim :

(b) Consider a program for determining the Previous Date. Its input is triple of Day, Month and Year with values in the range :

$$1 \le \text{Day} \le 31$$

$$1 \le \text{Month} \le 12$$

$$1900 \le \text{Year} \le 2025$$

Possible outputs would be Previous Date or Invalid Date. Design the Boundary Value Test Cases.

Algorithm :

- Take 3 inputs from the user for Day, Month and Year.
- Check whether they lie in the given intervals.
- If the condition is false, stop the program and exit.
- If the condition is true, calculate the date according to the given values.
- Subtract 1 day from it to get the Previous Date.
- According to the formula ***4n+1,*** there will be 13 test cases, where n is number of inputs.

Code :

```
#include <iostream>
using namespace std;
void bva(int, int, int);
int main()
{   int amin, amax, bmin, bmax, cmin, cmax;
    int x, y, z;
    cout << "Enter Range for Day : ";
    cin >> amin >> amax;
    cout << "Enter Range for Month : ";
    cin >> bmin >> bmax;
    cout << "Enter Range for Year : ";
    cin >> cmin >> cmax;
    if (amin < 1 || amax > 31)
    {
        cout << "Invalid Day Range";
        return 0;
    }
    if (bmin < 1 || bmax > 12)
    {
        cout << "Invalid Month Range";
        return 0;
    }
    if (cmin < 1900 || amax > 2025)
    {
        cout << "Invalid Year Range";
        return 0;
    }
    int anominal = (amin + amax) / 2;
    int avalues[] = {amin, amin + 1, anominal, amax - 1, amax};
    int bnominal = (bmin + bmax) / 2;
    int bvalues[] = {bmin, bmin + 1, bnominal, bmax - 1, bmax};
    int cnominal = (cmin + cmax) / 2;
    int cvalues[] = {cmin, cmin + 1, cnominal, cmax - 1, cmax};
    cout << "Day\tMonth\tYear\tExpected Output" << endl;
    for (int i = 0; i < 5; i++)
        bva(avalues[i], bnominal, cnominal);
    for (int i = 0; i < 5; i++)
        bva(anominal, bvalues[i], cnominal);
    for (int i = 0; i < 5; i++)
        bva(anominal, bnominal, cvalues[i]);
    cout << "Enter the Day, Month and Year : ";
    cin >> x >> y >> z;
    cout << "Day\tMonth\tYear\tExpected Output" << endl;
    bva(x, y, z);
    return 0;
}

void bva(int a, int b, int c)
{
    cout << a << "\t" << b << "\t" << c << "\t";
    if (a != 1)
    {
        if ((b == 2 || b == 4 || b == 6 || b == 9 || b == 11) & (a == 31))
            cout << "Invalid Date" << endl;
        else if ((b == 2) & (a == 30))
            cout << "Invalid Date" << endl;
        else if ((b == 2) & (a == 29) & (c % 4 != 0))
            cout << "Invalid Date" << endl;
        else
            cout << a - 1 << "-" << b << "-" << c << endl;
    }
    else
    {
        if (b == 3)
        {
            if (c % 4 == 0)
                a = 29;
            else
                a = 28;
        }
        cout << a << "-" << b - 1 << "-" << c << endl;
    }
}
```

Boundary Value Analysis :

Range : R [1, 31] [1, 12] [1900, 2025]

Domain : Minimum = 1, 1, 1900

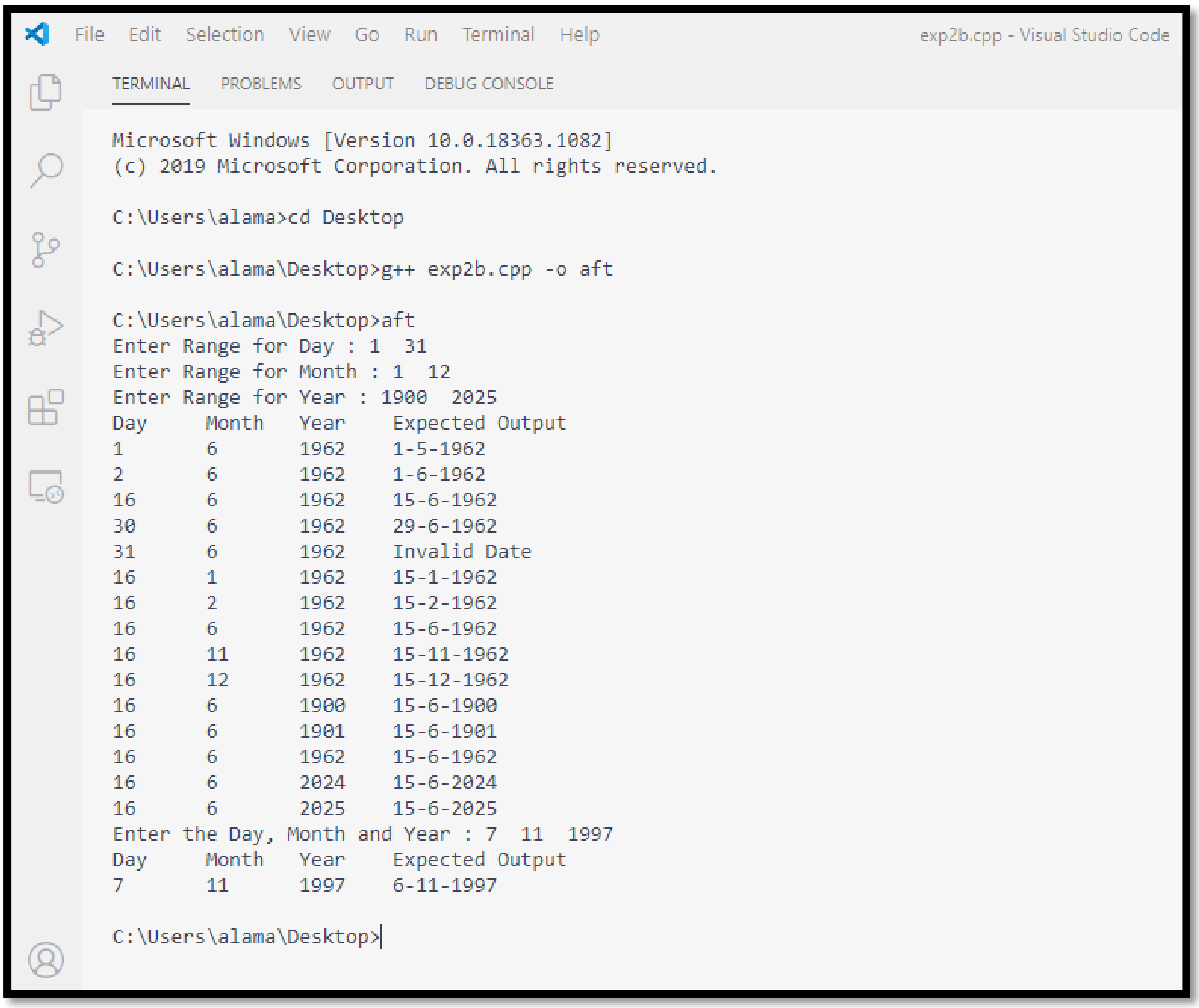
Above Minimum = 2, 2, 1901

Nominal = 16, 6, 1962

Below Maximum = 30, 11, 2024

Maximum = 31, 12, 2025

Output :



```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Desktop

C:\Users\alama\Desktop>g++ exp2b.cpp -o aft

C:\Users\alama\Desktop>aft
Enter Range for Day : 1  31
Enter Range for Month : 1  12
Enter Range for Year : 1900  2025
Day     Month   Year    Expected Output
1       6       1962    1-5-1962
2       6       1962    1-6-1962
16      6       1962    15-6-1962
30      6       1962    29-6-1962
31      6       1962    Invalid Date
16      1       1962    15-1-1962
16      2       1962    15-2-1962
16      6       1962    15-6-1962
16      11      1962    15-11-1962
16      12      1962    15-12-1962
16      6       1900    15-6-1900
16      6       1901    15-6-1901
16      6       1962    15-6-1962
16      6       2024    15-6-2024
16      6       2025    15-6-2025
Enter the Day, Month and Year : 7  11  1997
Day     Month   Year    Expected Output
7       11      1997    6-11-1997

C:\Users\alama\Desktop>
```

07/09/2020

Mohd. Aftab Alam
02013302717

EXPERIMENT 3

Aim :

(a) To determine the type of Triangle. Its input is triple of positive integers (say a, b, c) and values may be from the interval [1, 100]. The output may have one of the following :

Equilateral, Isosceles, Scalene or Not a Triangle.

Perform Boundary Value Analysis and show the Test Cases.

Algorithm :

- Take 3 inputs from the user for the sides of the Triangle.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- If the condition is true, check the type of Triangle.
- If all three sides are equal, Equilateral Triangle.
- If any two sides are equal, Isosceles Triangle.
- If all three sides are different, Scalene Triangle
- According to the formula ***4 n* + 1,** there will be ***13*** test cases, where n is number of inputs.

Code :

```
#include <iostream>

using namespace std;

void bva(int, int, int);

int main()
{
    int min, max;
    int x, y, z;
    cout << "Enter Range : ";
    cin >> min >> max;

    if (min < 0 || max > 100)
    {
        cout << "Invalid Range";
        return 0;
    }

    int nominal = (min + max) / 2;
    int values[] = {min, min + 1, nominal, max - 1, max};
    cout << "a\tb\tc\tOutput" << endl;

    for (int i = 0; i < 5; i++)
    {
        bva(values[i], nominal, nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, values[i], nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, nominal, values[i]);
    }

    cout << "Enter the Sides of Triangle (a, b, c) : ";
    cin >> x >> y >> z;
    cout << "a\tb\tc\tOutput" << endl;
    bva(x, y, z);

    return 0;
}

void bva(int a, int b, int c)
{
    cout << a << "\t" << b << "\t" << c << "\t";

    if (a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100)
        cout << "Invalid Range" << endl;

    else if ((a < b + c) && (b < a + c) && (c < a + b))
    {
        if ((a == b) && (b == c))
            cout << "Equilateral Triangle" << endl;
        else if ((a != b) && (b != c) && (c != a))
            cout << "Scalene Triangle" << endl;
        else
            cout << "Isosceles Triangle" << endl;
    }

    else
        cout << "Not a Triangle" << endl;
}
```

Boundary Value Analysis :

Range : R [1, 100]

Domain : Minimum = 1

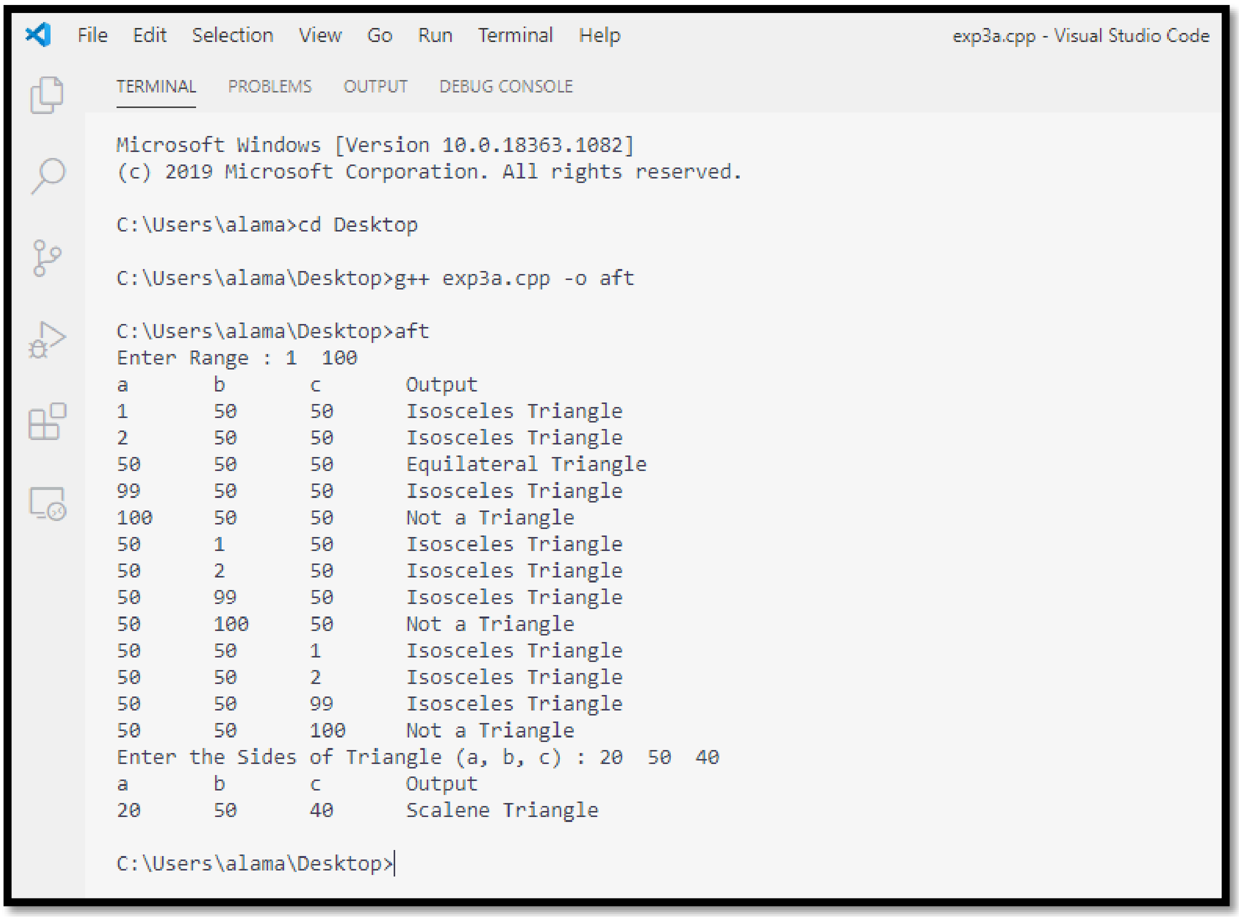
Above Minimum = 2

Nominal = 50

Below Maximum = 99

Maximum = 100

Output Screenshot :



```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Desktop

C:\Users\alama\Desktop>g++ exp3a.cpp -o aft

C:\Users\alama\Desktop>aft
Enter Range : 1  100
a       b       c       Output
1       50      50      Isosceles Triangle
2       50      50      Isosceles Triangle
50      50      50      Equilateral Triangle
99      50      50      Isosceles Triangle
100     50      50      Not a Triangle
50      1       50      Isosceles Triangle
50      2       50      Isosceles Triangle
50      99      50      Isosceles Triangle
50      100     50      Not a Triangle
50      50      1       Isosceles Triangle
50      50      2       Isosceles Triangle
50      50      99      Isosceles Triangle
50      50      100     Not a Triangle
Enter the Sides of Triangle (a, b, c) : 20  50  40
a       b       c       Output
20      50      40      Scalene Triangle

C:\Users\alama\Desktop>
```

Aim :

(b) Write a program for classification of Triangle on the basis of angle. Its input is triple of positive integers (say a, b, c) and values may be from the interval [1, 100]. The output may have one of the following :

Right Angled, Acute Angled, Obtuse Angled or Not a Triangle.

Perform Boundary Value Analysis and show the Robust Test Cases.

Algorithm :

- Take 3 inputs from the user for the sides of the Triangle.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- If the condition is true, check the type of Triangle.
- If the square of one side is equal to the sum of squares of other two sides, Right Angled Triangle.
- Else if the square of one side is greater than the sum of squares of other two sides, Obtuse Angled.
- Else if the square of one side is less than the sum of squares of other two sides, Acute Angled.
- According to the formula ***4 n + 1,*** there will be ***13*** test cases, where n is number of inputs.
- Robust Test Cases ***= 6 n + 1 = 6 * 3 + 1 = 18 + 1 = 19.***

Code :

```
#include <iostream>

using namespace std;

void bva(int, int, int);

int main()
{
    int min, max;
    int x, y, z;
    cout << "Enter Range : ";
    cin >> min >> max;

    if (min < 0 || max > 100)
    {
        cout << "Invalid Range";
        return 0;
    }

    int nominal = (min + max) / 2;
    int values[] = {min, min + 1, nominal, max - 1, max};
    cout << "a\tb\tc\tOutput" << endl;

    for (int i = 0; i < 5; i++)
    {
        bva(values[i], nominal, nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, values[i], nominal);
    }

    for (int i = 0; i < 5; i++)
    {
        if (values[i] != nominal)
            bva(nominal, nominal, values[i]);
    }

    cout << "Enter the Sides of Triangle (a, b, c) : ";
    cin >> x >> y >> z;
    cout << "a\tb\tc\tOutput" << endl;
    bva(x, y, z);

    return 0;
}

void bva(int a, int b, int c)
{
    cout << a << "\t" << b << "\t" << c << "\t";

    if (a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100)
        cout << "Invalid Range" << endl;

    else if ((a < b + c) && (b < a + c) && (c < a + b))
    {
        if ((a * a == b * b + c * c) || (b * b == c * c + a * a)
        || (c * c == a * a + b * b))
            cout << "Right Angled Triangle" << endl;
        else if ((a * a > b * b + c * c) || (b * b > c * c + a * a)
        || (c * c > a * a + b * b))
            cout << "Obtuse Angled Triangle" << endl;
        else
            cout << "Acute Angled Triangle" << endl;
    }
    else
        cout << "Not a Triangle" << endl;
}
```

Mohd. Aftab Alam
02013302717

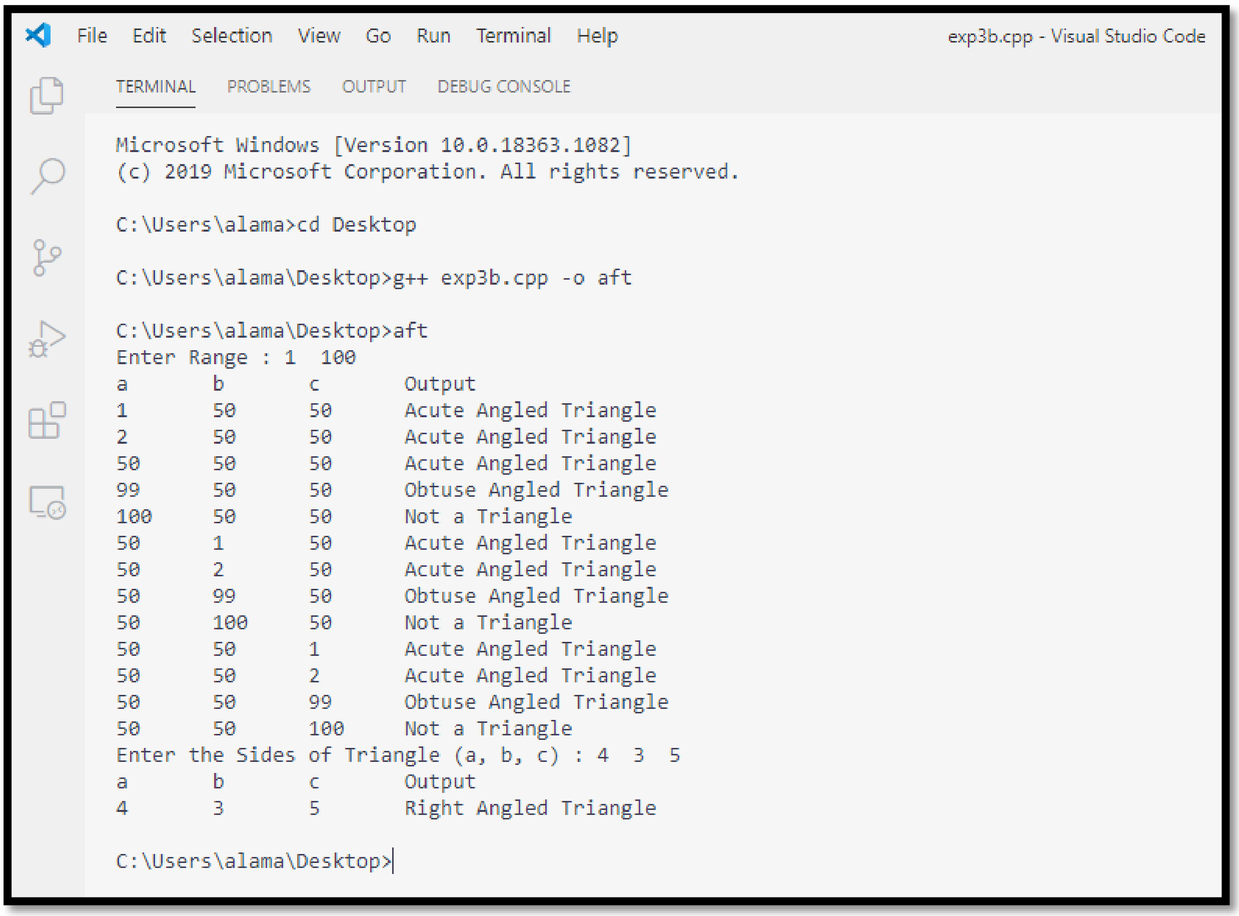
Boundary Value Analysis :

Range : R [1, 100]

Domain :

Below Minimum	0
Minimum	1
Above Minimum	2
Nominal	50
Below Maximum	99
Maximum	100
Above Maximum	101

Output Screenshot :



```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Desktop

C:\Users\alama\Desktop>g++ exp3b.cpp -o aft

C:\Users\alama\Desktop>aft
Enter Range : 1  100
a       b       c       Output
1       50      50      Acute Angled Triangle
2       50      50      Acute Angled Triangle
50      50      50      Acute Angled Triangle
99      50      50      Obtuse Angled Triangle
100     50      50      Not a Triangle
50      1       50      Acute Angled Triangle
50      2       50      Acute Angled Triangle
50      99      50      Obtuse Angled Triangle
50      100     50      Not a Triangle
50      50      1       Acute Angled Triangle
50      50      2       Acute Angled Triangle
50      50      99      Obtuse Angled Triangle
50      50      100     Not a Triangle
Enter the Sides of Triangle (a, b, c) : 4  3  5
a       b       c       Output
4       3       5       Right Angled Triangle

C:\Users\alama\Desktop>
```

Mohd. Aftab Alam
02013302717

Robust Test Cases :

Test Case	**a**	**b**	**c**	**Expected Output**
1	1	50	50	Acute Angled Triangle
2	2	50	50	Acute Angled Triangle
3	50	50	50	Acute Angled Triangle
4	99	50	50	Obtuse Angled Triangle
5	100	50	50	Not a Triangle
6	0	50	50	Invalid Range
7	101	50	50	Invalid Range
8	50	1	50	Acute Angled Triangle
9	50	2	50	Acute Angled Triangle
10	50	99	50	Obtuse Angled Triangle
11	50	100	50	Not a Triangle
12	50	0	50	Invalid Range
13	50	101	50	Invalid Range
14	50	50	1	Acute Angled Triangle
15	50	50	2	Acute Angled Triangle
16	50	50	99	Obtuse Angled Triangle
17	50	50	100	Not a Triangle
18	50	50	0	Invalid Range
19	50	50	101	Invalid Range

14/09/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 4

Aim :

(a) Write a Program in C/C++ to determine the area of Circle, Triangle, Square and Rectangle. Values may be from the interval [1, 100] and perform Equivalence Class Testing.

Algorithm :

- Take inputs from the user according to Polygon.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- If the condition is true, calculate the area of Polygon.
    - If Circle, ***area = π * radius²***
    - If Triangle, ***area = ½ * base * height***
    - If Square, ***area = side²***
    - If Rectangle, ***area = length * breadth***
- Perform equivalence class testing accordingly.

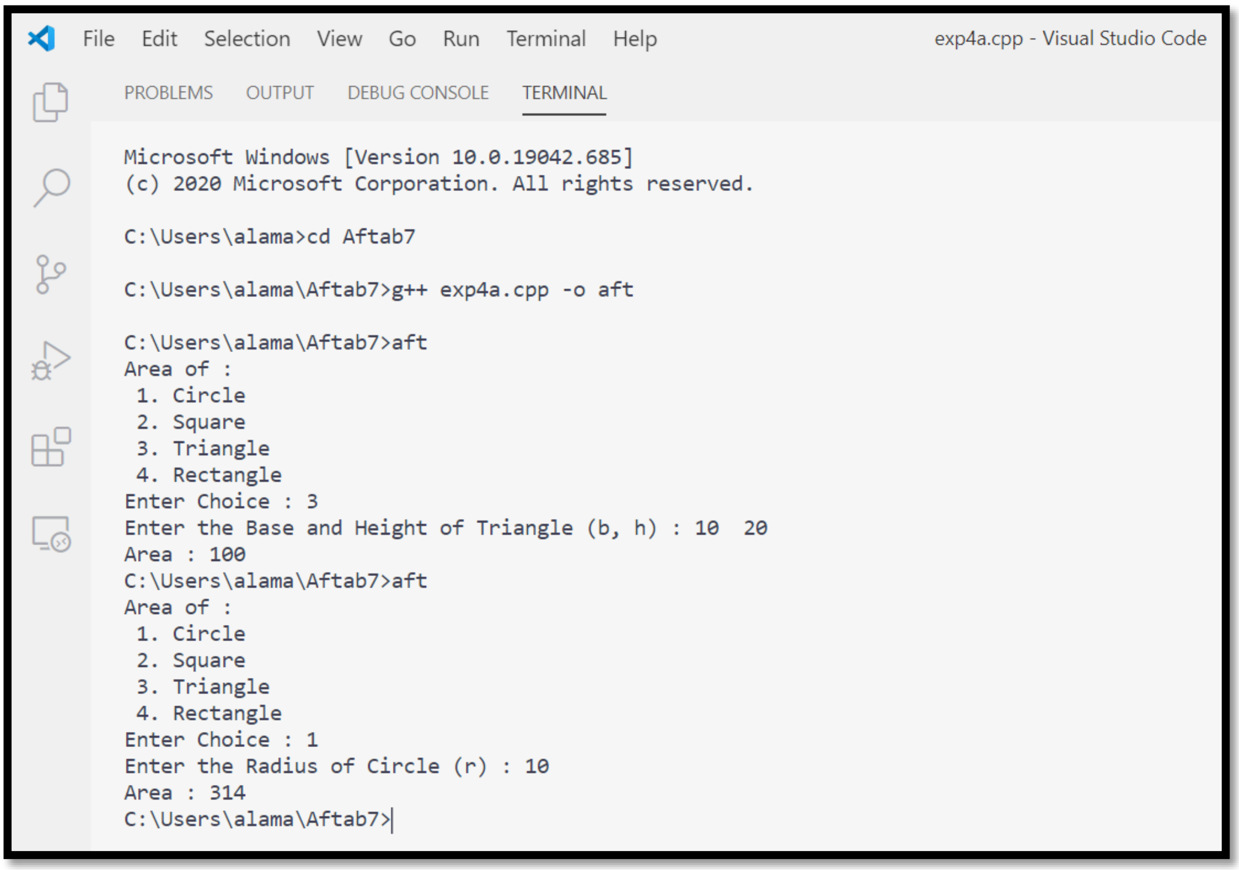
Code :

```
#include <iostream>
using namespace std;
float circle()
{
    float r;
    cout << "Enter the Radius of Circle (r) : ";
    cin >> r;
    if (r < 1 || r > 100)
    {   cout << "Out of Range";
        return 0; }
    float area = 3.14 * r * r;
    return area;
}
float triangle()
{
    float b, h;
    cout << "Enter the Base and Height of Triangle (b, h) : ";
    cin >> b >> h;
    if (b < 1 || b > 100 || h < 1 || h > 100)
    {   cout << "Out of Range";
        return 0; }
    float area = 0.5 * b * h;
    return area;
}
```

```
float square()
{   float s;
    cout << "Enter the Side of Square (s) : ";
    cin >> s;
    if (s < 1 || s > 100)
    {   cout << "Out of Range";
        return 0; }
    float area = s * s;
    return area; }
float rectangle()
{   float l, b;
    cout << "Enter the Length and Breadth of Rectangle (l, b) : ";
    cin >> l >> b;
    if (l < 1 || l > 100 || b < 1 || b > 100)
    {   cout << "Out of Range";
        return 0; }
    float area = l * b;
    return area; }
int main()
{   int ch;
    float area;
    cout << "Area of : \n 1. Circle \n 2. Square \n 3. Triangle
             \n 4. Rectangle";
    cout << "\nEnter Choice : ";
    cin >> ch;
    switch (ch)
    {
    case 1:
        area = circle();
        cout << "Area : " << area;
        break;
    case 2:
        area = square();
        cout << "Area : " << area;
        break;
    case 3:
        area = triangle();
        cout << "Area : " << area;
        break;
    case 4:
        area = rectangle();
        cout << "Area : " << area;
        break;
    default:
        cout << "Wrong Choice";
    }
    return 0;
}
```

14/09/2020 **Mohd. Aftab Alam**
02013302717

Output Screenshots :

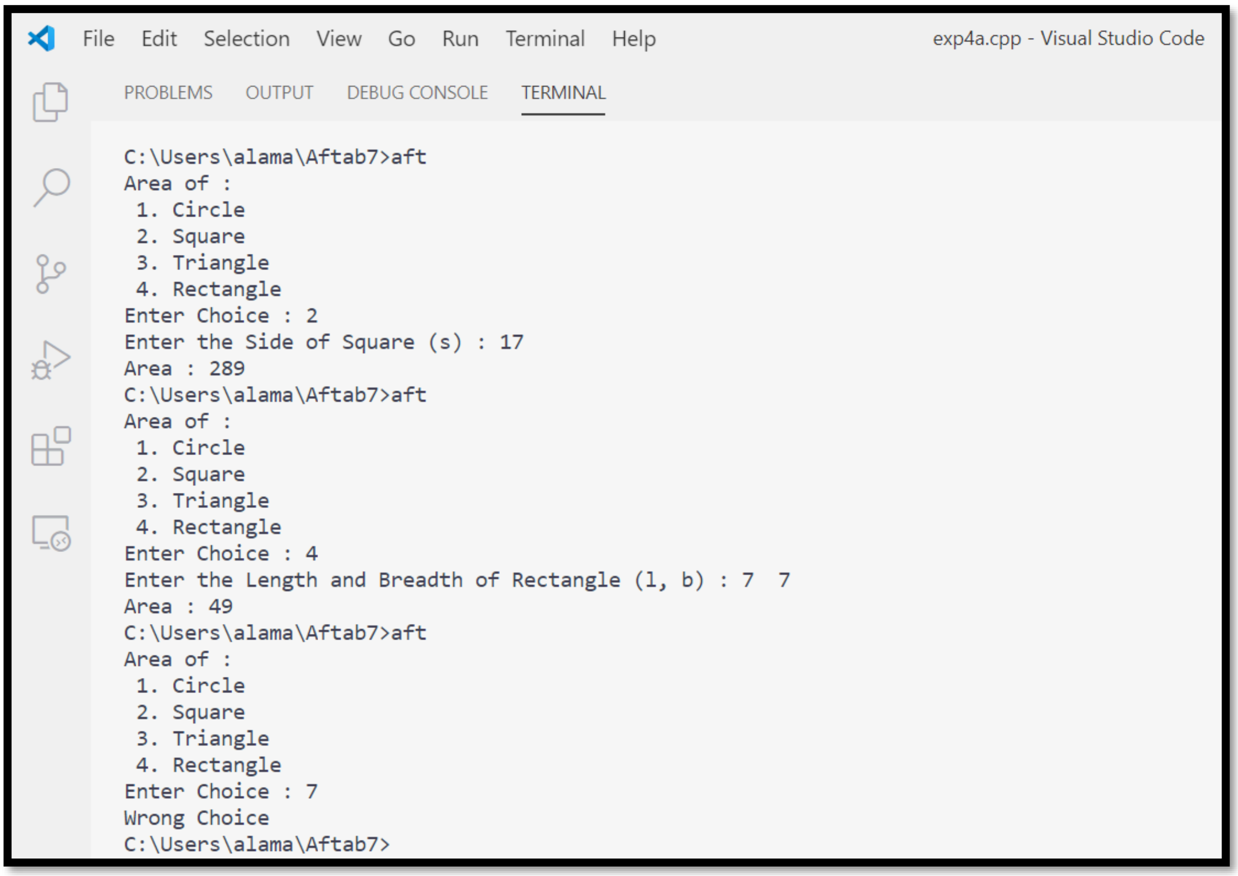


```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp4a.cpp -o aft

C:\Users\alama\Aftab7>aft
Area of :
 1. Circle
 2. Square
 3. Triangle
 4. Rectangle
Enter Choice : 3
Enter the Base and Height of Triangle (b, h) : 10  20
Area : 100
C:\Users\alama\Aftab7>aft
Area of :
 1. Circle
 2. Square
 3. Triangle
 4. Rectangle
Enter Choice : 1
Enter the Radius of Circle (r) : 10
Area : 314
C:\Users\alama\Aftab7>
```



```
C:\Users\alama\Aftab7>aft
Area of :
 1. Circle
 2. Square
 3. Triangle
 4. Rectangle
Enter Choice : 2
Enter the Side of Square (s) : 17
Area : 289
C:\Users\alama\Aftab7>aft
Area of :
 1. Circle
 2. Square
 3. Triangle
 4. Rectangle
Enter Choice : 4
Enter the Length and Breadth of Rectangle (l, b) : 7  7
Area : 49
C:\Users\alama\Aftab7>aft
Area of :
 1. Circle
 2. Square
 3. Triangle
 4. Rectangle
Enter Choice : 7
Wrong Choice
C:\Users\alama\Aftab7>
```

14/09/2020 **Mohd. Aftab Alam**
02013302717

Equivalence Class Testing :

Range : R [1, 200]

Case 1 : CIRCLE

Input Domain :

I1 = {r: r < = 0}

I2 = {r: r > 200}

I3 = {r: 1 < = r < = 200}

Output Domain :

O1 = {: Circle if 1 < = r < = 200}

O2 = {: Not a Circle if r < = 0}

Case 2 : SQUARE

Input Domain :

I1 = {s: s < = 0}

I2 = {s: s > 200}

I3 = {s: 1 < = s < = 200}

Output Domain :

O1 = {: Square if s > 0}

O2 = {: Not a Square if s < = 0}

Case 3 : TRIANGLE

Input Domain :

I1 = {h: h < = 0}

I2 = {h: h > 200}

I3 = {h: 1 < = h < = 200}

I4 = {b: b < = 0}

I5 = {b: b > 200}

I6 = {b: 1 < = b < = 200}

Output Domain :

O1 = {: Triangle if h > 0, b > 0}

O2 = {: Not a Triangle if h < = 0, b < = 0}

Case 4 : RECTANGLE

Input Domain :

I1 = {l: l < = 0}

I2 = {l: l > 200}

I3 = {l: 1 < = l < = 200}

I4 = {b: b < = 0}

I5 = {b: b > 200}

I6 = {b: 1 < = b < = 200}

Output Domain :

O1 = {: Rectangle if l > 0, b > 0}

O2 = {: Not a Rectangle if l < = 0, b < = 0}

14/09/2020 **Mohd. Aftab Alam**
02013302717

Circle Test Cases :

Test Case	r	Expected Output
1	0	Invalid Input
2	100	31400
3	201	Invalid Input

Square Test Cases :

Test Case	s	Expected Output
1	0	Invalid Input
2	100	10000
3	201	Invalid Input

Triangle Test Cases :

Test Case	h	b	Expected Output
1	0	100	Invalid Input
2	100	100	5000
3	201	100	Invalid Input
4	100	0	Invalid Input
5	100	100	5000
6	100	201	Invalid Input

Rectangle Test Cases :

Test Case	l	b	Expected Output
1	0	100	Invalid Input
2	100	100	10000
3	201	100	Invalid Input
4	100	0	Invalid Input
5	100	100	10000
6	100	201	Invalid Input

Mohd. Aftab Alam
02013302717

Aim :

(b) Write a Program in C/C++ to determine the type of Triangle that is Equilateral, Isosceles, Scalene or Not a Triangle. Values may be from the interval [1, 100] and perform Decision Table Based Testing and Equivalence Class Testing.

Algorithm :

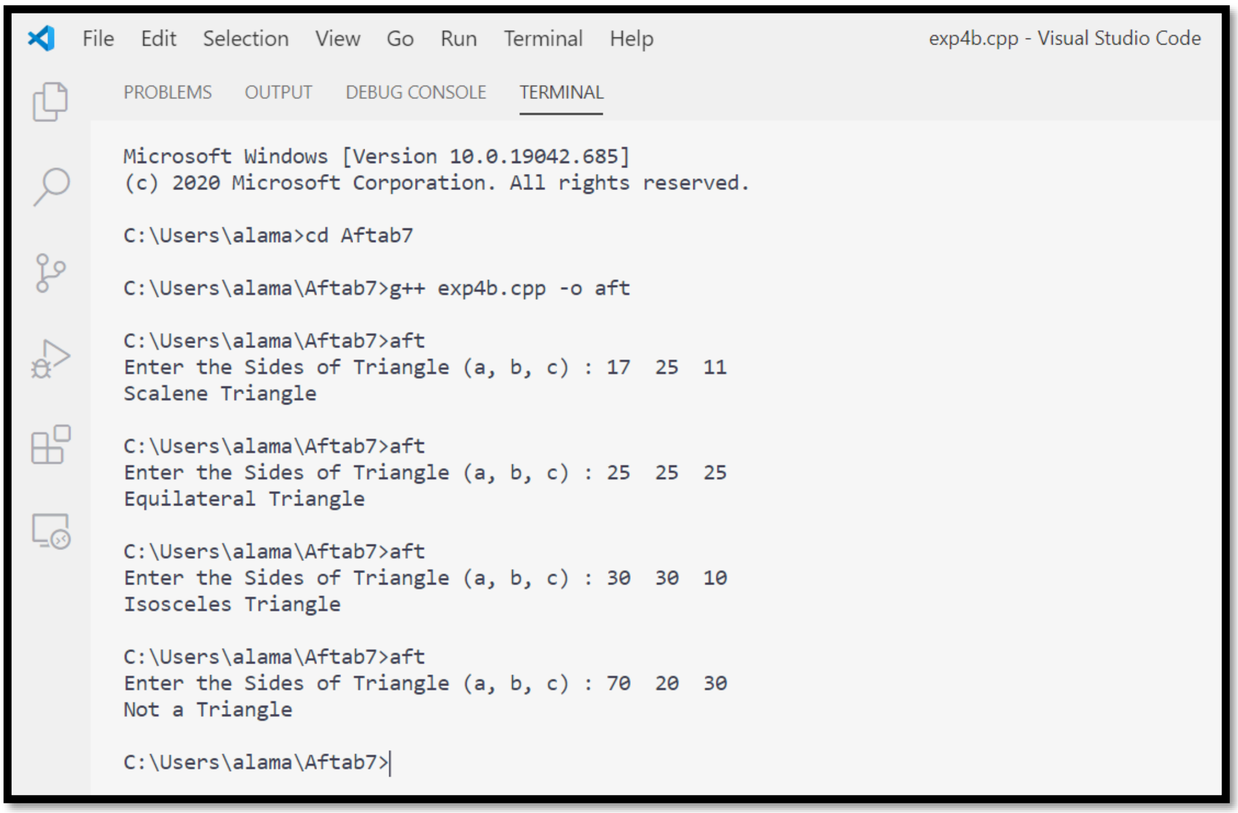
- Take 3 inputs from the user for the sides of the Triangle.
- Check whether they lie in the given interval.
- If the condition is false, stop the program and exit.
- If the condition is true, check the type of Triangle.
    - If all three sides are equal, Equilateral Triangle.
    - If any two sides are equal, Isosceles Triangle.
    - If all three sides are different, Scalene Triangle.
- Make decision table according to the output.
- Perform equivalence class testing accordingly.

Code :

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a, b, c;
    cout << "Enter the Sides of Triangle (a, b, c) : ";
    cin >> a >> b >> c;
    if (a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100)
        cout << "Out of Range" << endl;
    else if ((a < b + c) && (b < a + c) && (c < a + b))
    {
        if ((a == b) && (b == c))
            cout << "Equilateral Triangle" << endl;
        else if ((a != b) && (b != c) && (c != a))
            cout << "Scalene Triangle" << endl;
        else
            cout << "Isosceles Triangle" << endl;
    }
    else
        cout << "Not a Triangle" << endl;
    return 0;
}
```

Output Screenshot :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp4b.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter the Sides of Triangle (a, b, c) : 17  25  11
Scalene Triangle

C:\Users\alama\Aftab7>aft
Enter the Sides of Triangle (a, b, c) : 25  25  25
Equilateral Triangle

C:\Users\alama\Aftab7>aft
Enter the Sides of Triangle (a, b, c) : 30  30  10
Isosceles Triangle

C:\Users\alama\Aftab7>aft
Enter the Sides of Triangle (a, b, c) : 70  20  30
Not a Triangle

C:\Users\alama\Aftab7>
```

Decision Table Based Testing :

Range : R [1, 100]

c1: a < b + c ?	F	T	T	T	T	T	T	T	T	T	T
c2: b < a + c ?	—	F	T	T	T	T	T	T	T	T	T
c3: c < a + b ?	—	—	F	T	T	T	T	T	T	T	T
c4: a = b ?	—	—	—	T	T	T	T	F	F	F	F
c5: a = c ?	—	—	—	T	T	F	F	T	T	F	F
c6: b = c ?	—	—	—	T	F	T	F	T	F	T	F
a1: Not a Triangle	x	x	x								
a2: Scalene Triangle											x
a3: Isosceles Triangle						x			x	x	
a4: Equilateral Triangle				x							
a5: Impossible					x	x		x			

Corresponding Test Cases :

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral Triangle
DT5	?	?	?	Impossible
DT6	?	?	?	Impossible
DT7	2	2	3	Isosceles Triangle
DT8	?	?	?	Impossible
DT9	2	3	2	Isosceles Triangle
DT10	3	2	2	Isosceles Triangle
DT11	3	4	5	Scalene Triangle

Equivalence Class Testing :

Range : R [1, 100]

Input Domain :

I1 = {0 < a < = 10}	I11 = {a = b, b ! = c}
I2 = {a < 0}	I12 = {b = c, c ! = a}
I3 = {a > 10}	I13 = {a = c, c ! = b}
I4 = {0 < b < = 10}	I14 = {a ! = b ! = c}
I5 = {b < 0}	I15 = {a + b = c}
I6 = {b > 10}	I16 = {a + b < c}
I7 = {0 < c < = 10}	I17 = {b + c = a}
I8 = {c < 0}	I18 = {b + c < a}
I9 = {c > 10}	I19 = {c + a = b}
I10 = {a = b = c}	I20 = {c + a > b}

Output Domain :

O1 = Not a Triangle

O2 = Equilateral Triangle

O3 = Isosceles Triangle

O4 = Scalene Triangle

Test Cases :

Test Case	a	b	c	Expected Output	Actual Output
O1	10	5	5	Not a Triangle	Not a Triangle
O2	5	5	5	Equilateral Triangle	Equilateral Triangle
O3	1	5	5	Isosceles Triangle	Isosceles Triangle
O4	10	9	5	Scalene Triangle	Scalene Triangle
I1	5	5	5	Equilateral Triangle	Equilateral Triangle
I2	0	5	5	Invalid Input	Invalid Input
I3	11	5	5	Invalid Input	Invalid Input
I4	5	5	5	Equilateral Triangle	Equilateral Triangle
I5	5	0	5	Invalid Input	Invalid Input
I6	5	11	5	Invalid Input	Invalid Input
I7	5	5	5	Equilateral Triangle	Equilateral Triangle
I8	5	5	0	Invalid Input	Invalid Input
I9	5	5	11	Invalid Input	Invalid Input
I10	5	5	5	Equilateral Triangle	Equilateral Triangle
I11	5	5	1	Isosceles Triangle	Isosceles Triangle
I12	1	5	5	Isosceles Triangle	Isosceles Triangle
I13	5	1	5	Isosceles Triangle	Isosceles Triangle
I14	9	5	10	Scalene Triangle	Scalene Triangle
I15	5	5	10	Not a Triangle	Not a Triangle
I16	1	5	10	Not a Triangle	Not a Triangle
I17	10	5	5	Not a Triangle	Not a Triangle
I18	10	5	1	Not a Triangle	Not a Triangle
I19	5	10	5	Not a Triangle	Not a Triangle
I20	5	10	1	Not a Triangle	Not a Triangle

22/10/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 5

Aim :

(a) Write a Program in C/C++ to compute previous date, given the present date as input and perform data flow testing.

1. Algorithm
2. Define/Use Table
3. Du Path Table
4. All uses-definition clear
5. Test case table of all uses
6. Test case table of all definition

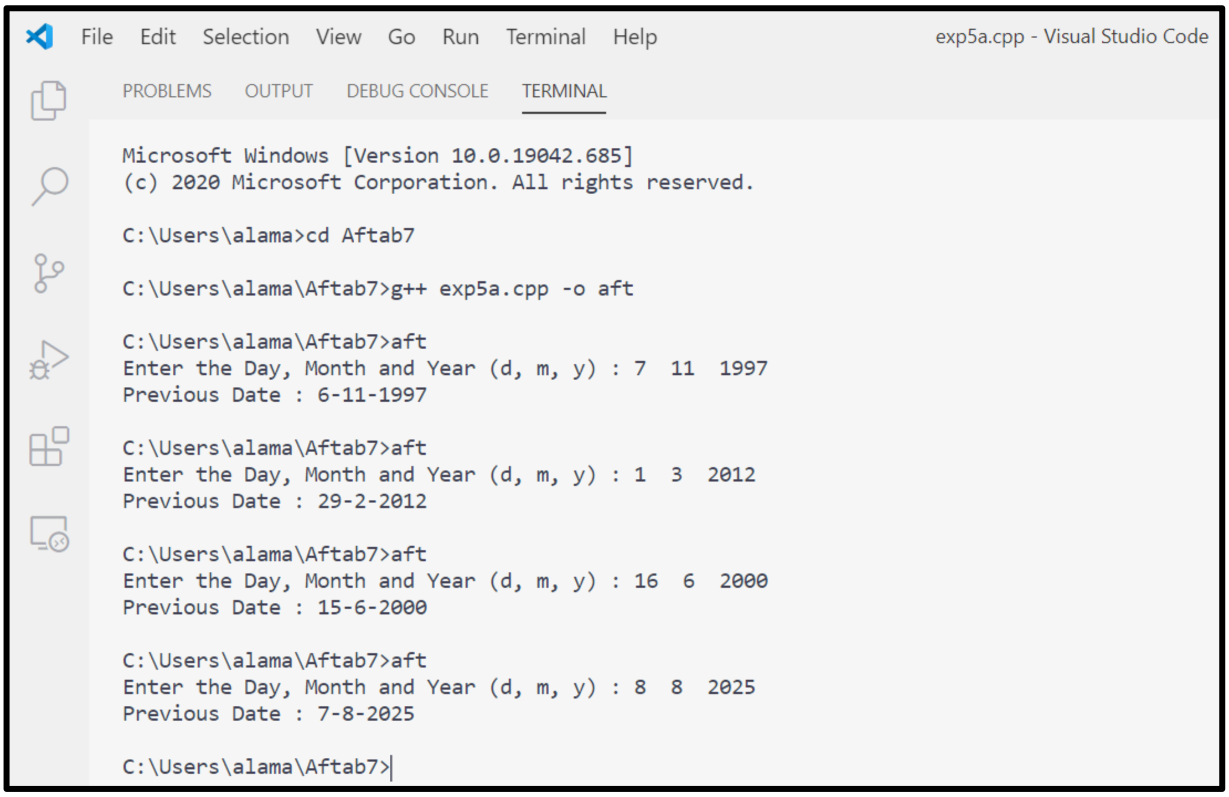
Algorithm :

- Take 3 inputs from the user for Day, Month and Year.
- Check whether they lie in the given intervals.
- If the condition is false, stop the program and exit.
- If the condition is true, calculate the date according to the given values.
- Subtract 1 day from it to get the Previous Date.

Code :

```
#include <iostream>
using namespace std;
int main()
{   int d, m, y;
    cout << "Enter the Day, Month and Year (d, m, y) : ";
    cin >> d >> m >> y;
    if (d != 1)
    {   if ((m == 2 || m == 4 || m == 6 || m == 9 || m == 11) & (d == 31))
            cout << "Invalid Date";
        else if ((m == 2) & (d == 30))
            cout << "Invalid Date";
        else if ((m == 2) & (d == 29) & (y % 4 != 0))
            cout << "Invalid Date";
        else
        cout << "Previous Date : " << d - 1 << "-" << m << "-" << y; }
    else
    {
        if (m == 3)
        {
            if (y % 4 == 0)
                d = 29;
            else
                d = 28;
        }
        cout << "Previous Date : " << d << "-" << m - 1 << "-" << y;
    }
    return 0;
}
```

Output Screenshots :


File Edit Selection View Go Run Terminal Help
exp5a.cpp - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp5a.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter the Day, Month and Year (d, m, y) : 7  11  1997
Previous Date : 6-11-1997

C:\Users\alama\Aftab7>aft
Enter the Day, Month and Year (d, m, y) : 1  3  2012
Previous Date : 29-2-2012

C:\Users\alama\Aftab7>aft
Enter the Day, Month and Year (d, m, y) : 16  6  2000
Previous Date : 15-6-2000

C:\Users\alama\Aftab7>aft
Enter the Day, Month and Year (d, m, y) : 8  8  2025
Previous Date : 7-8-2025

C:\Users\alama\Aftab7>
```

Define/Use Table :

S. No.	Variable	Defined at Node	Used at Node
1	D	3,16,20,22,25,29,32,36	5,10,13,36,38
2	M	3,17,23,26,30,33	7,10.15,17,18,23,24,28,30,33,38
3	Y	3,27	9,10,19,27,38

Du Path Table :

S. No.	Variable	Du Path (Begin, End)		
1	D	3,5	16,5	20,5
		3,10	16,10	20,10
		3,13	16,13	20,13
		3,36	16,36	20,36
		3,38	16,38	20,38
		22,5	25,5	29,5
		22,10	25,10	29,10
		22,13	25,13	29,13
		22,36	25,36	29,36
		22,38	25,38	29,38
		32,5	32,36	36,10
		32,10	32,38	36,13
		32,13	36,5	36,36
				36,38
2	M	3, 7	17, 7	23, 7
		3,10	17,10	23,10
		3,15	17,15	23,15
		3,17	17,17	23,17
		3,18	17,18	23,18
		3,23	17,23	23,23
		3,24	17,24	23,24
		3,28	17,28	23,28
		3,30	17,30	23,30
		3,33	17,33	23,33
		3,38	17,38	23,38
		26, 7	30, 7	33, 7
		26,10	30,10	33,10
		26,15	30,15	33,15
		26,17	30,17	33,17
		26,18	30,18	33,18
		26,23	30,23	33,23
		26,24	30,24	33,24
		26,28	30,28	33,28
		26,30	30,30	33,30
		26,33	30,33	33,33
		26,38	30,38	33,38
3	Y	3,9	3,27	27,10
		3,10	3,38	27,19
		3,19	27,9	27,27
				27,38

All uses-definition clear :

S. No.	Variable	Du Path (Begin, End)	Definition clear?
1	D	3,5	Yes
2	D	3,10	Yes
3	D	3,10,13	Yes
4	D	3,10,13,36	No
5	D	3,10,13,36,38	No
6	D	(16,5), (16,10), (16,13), (16,36)	Not Possible
7	D	16,17,38	Yes
8	D	(20,5), (20,10), (20,13), (20,36)	Not Possible
9	D	20,23,38	Yes
10	D	(22,5), (22,10), (22,13), (22,36)	Not Possible
11	D	22,23,38	Yes
12	D	(25,5), (25,10), (25,13), (25,36)	Not Possible
13	D	25,26,27,38	Yes
14	D	(29,5), (29,10), (29,13), (29,36)	Not Possible
15	D	29,30,38	Yes
16	D	(32,5), (32,10), (32,13), (32,36)	Not Possible
17	D	32,33,38	Yes
18	D	(36,5), (36,10), (36,13)	Not Possible
19	D	36,36	No
20	D	36,37,38	Yes
21	M	3,7	Yes
22	M	3,7,10	Yes
23	M	3,10,13,15	Yes
24	M	3,10,13,15,17	No
25	M	3,10,13,15,18	Yes
26	M	3,10,15,18,19,22,23	No
27	M	3,10,15,18,24	Yes
28	M	3,10,15,18,24,28	Yes
29	M	3,10,15,18,24,28,29,30	No
30	M	3,10,15,18,24,28,32,33	No
31	M	3,10,15,18,24,28,32,33,38	No
32	M	(17,7),(17,10),(17,15),(17,18),(17,23),(17,24),(17,28),(17,30),(17,3)	Not Possible
33	M	17,17	No
34	M	17,38	Yes
35	M	(23,7),(23,10),(23,15),(23,17),(23,18),(23,23),(23,24),(23,28),(23,30),(23,33)	Not Possible
36	M	23,38	Yes
37	M	(26,7),(26,10),(26,15),(26,17),(26,18),(26,23),(26,24),(26,28),(26,30),(26,33)	Not Possible
38	M	26,27,38	Yes
39	M	(30,7),(30,10),(30,15),(30,17),(30,18),(30,23),(30,24),(30,28),(30,33)	Not Possible
40	M	30,30	No
41	M	30,38	Yes
42	M	(33,7),(33,10),(33,15),(33,17),(33,18),(33,23),(33,24),(33,28)	Not Possible
43	M	33,33	No
44	M	33,38	Yes
45	Y	3,9	Yes
46	Y	3,9,10	Yes
47	Y	3,10,13,15,18,19	Yes
48	Y	3,10,13,15,18,24,27	No
49	Y	3,10,13, 15,18,24,27,38	No
50	Y	(27,9),(27,10),(27,19)	Not Possible
51	Y	27,27	No
52	Y	27,38	Yes

22/10/2020 **Mohd. Aftab Alam**
02013302717

Test case table of all users :

S. No.	Variables			Expected Output	Du Path
	D	M	Y		
1	15	1	1962	14/1/1962	3,5
2	15	1	1962	14/1/1962	3,10
3	1	6	1964	31/5/1964	3,10,13
4	1	6	1964	31/5/1964	16,17,38
5	1	3	1964	29/2/1964	20,23,38
6	1	3	1964	28/2/1962	22,23,38
7	1	1	1960	31/12/1959	25,26,27,38
8	1	2	1962	31/1/1962	29,30,38
9	1	5	1960	30/4/1960	32,33,38
10	12	4	1960	11/4/1960	36,37,38
11	15	1	1962	14/1/1962	3,7
12	15	1	1962	14/1/1962	3,7,10
13	1	6	2000	31/5/2000	3,10,13,15
14	1	3	2000	29/2/2000	3,10,13,15,18
15	1	1	2000	31/12/1999	3,10,13,15,18,24
16	1	2	2000	31/1/2000	3,10,13,15,18,24,28
17	1	6	2000	31/5/2000	17,38
18	1	3	2000	29/2/2000	23,38
19	1	1	2000	31/12/1999	26,27,38
20	1	2	2000	31/1/2000	30,38
21	1	7	2001	30/6/2001	33,38
22	15	1	1962	14/1/1962	3,9
23	15	1	1962	14/1/1962	3,9,10
24	1	3	2015	28/2/2015	3,10,13,15,18,19
25	1	1	2015	31/12/2014	27,38

Test case table of all definition :

S. No.	Variables			Expected Output	Du Path
	D	M	Y		
1	15	6	1972	14/6/1972	3,10,13,36
2	15	6	1972	14/6/1972	3,10,13,36,38
3	15	6	1972	14/6/1972	36,36
4	1	4	2012	31/3/2012	3,10,13,15,17
5	1	3	2011	28/2/2011	3,10,15,18,19,22,23
6	1	2	2012	31/1/2012	3,10,15,18,24,28,29,30
7	1	5	2012	30/4/2012	3,10,15,18,24,28,32,33
8	1	5	2012	30/4/2012	3,10,15,18,24,28,32,33,38
9	1	4	2012	31/3/2012	17,17
10	1	2	2012	31/1/2012	30,30
11	1	5	2012	30/4/2012	33,33
12	1	1	2013	31/12/2012	3,10,13,15,18,24,27
13	1	1	2013	31/12/2012	3,10,13, 15,18,24,27,38
14	1	1	2013	31/12/2012	27,27

22/10/2020 **Mohd. Aftab Alam**
02013302717

Aim : (b) Write a Program in C/C++ to compute **a^b** and perform data flow testing.

1. Algorithm
2. Define/Use Table
3. Du Path Table
4. All uses-definition clear
5. Test case table of all uses
6. Test case table of all definition

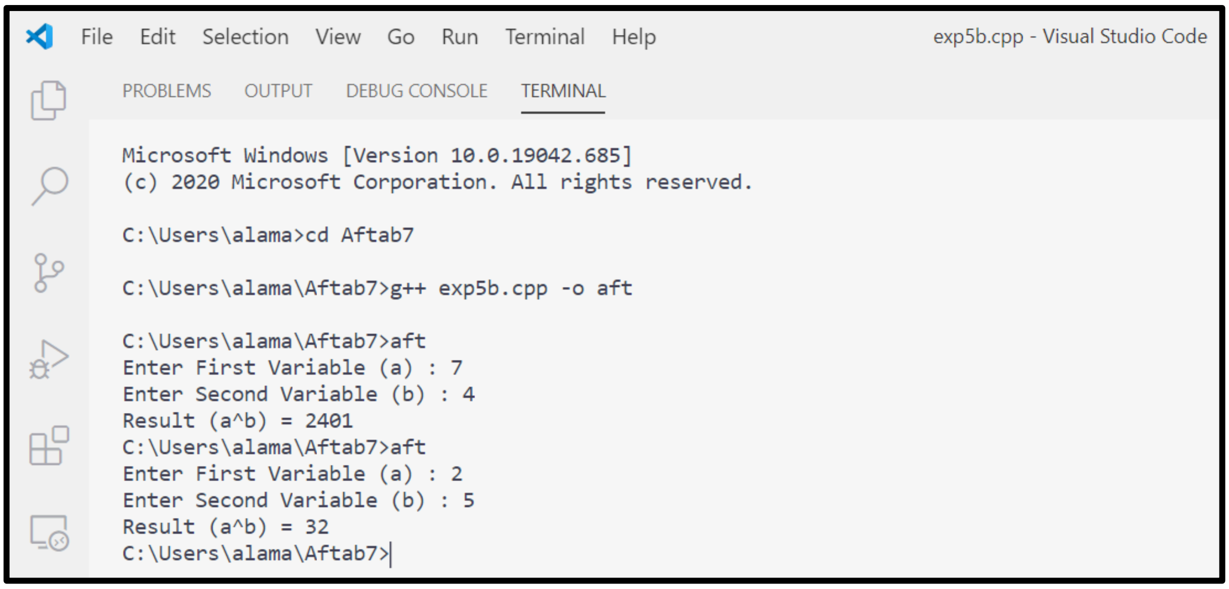
Algorithm :

- Take two inputs from the user for a and b.
- Calculate **a^b** using the pow() function.
- Store the result in another variable result.
- Print the result on the screen.

Code :

```cpp
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    float a, b, result;
    cout << "Enter First Variable (a) : ";
    cin >> a;
    cout << "Enter Second Variable (b) : ";
    cin >> b;
    result = pow(a, b);
    cout << "Result (a^b) = " << result;
    return 0;
}
```

Output Screenshot :


File Edit Selection View Go Run Terminal Help
exp5b.cpp - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp5b.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter First Variable (a) : 7
Enter Second Variable (b) : 4
Result (a^b) = 2401
C:\Users\alama\Aftab7>aft
Enter First Variable (a) : 2
Enter Second Variable (b) : 5
Result (a^b) = 32
C:\Users\alama\Aftab7>
```

Define/Use Table :

S. No.	Variable	Defined at Node	Used at Node
1	a	3	5,8
2	b	3	7,8
3	result	3,8	9

Du Path Table :

S. No.	Variable	Du Path (Begin, End)
1	a	3,5 3,8
2	b	3,7 3,8
3	result	3,9 8,9

All uses definition clear :

S. No.	Variable	Du Path (Begin, End)	Definition clear?
1	a	3,5	Yes
2	a	3,5,8	Yes
3	b	3,7	Yes
4	b	3,7,8	Yes
5	result	3,8,9	No
6	result	8,9	Yes

There is a total of 6 du-paths out of which 1 path is not defined clearly.

Test case table of all users :

S. No.	Variables			Expected Output	Du Path
	a	b	result		
1	2.5	4	39.0625	39.0625	3,5
2	2.5	4	39.0625	39.0625	3,5,8
3	4.3	5	1470.08	1470.08	3,7
4	4.3	5	1470.08	1470.08	3,7,8
5	5	2.2	34.4932	34.4932	8,9

Test case table of all definition :

S. No.	Variables			Expected Output	Du Path
	a	b	result		
1	5	2.2	34.4932	34.4932	3,8,9

EXPERIMENT 6

Aim : (a) Write a Program in C/C++ to compute total salary of an employee when his basic salary is given. (Given: HRA = 3% of basic, DA = 8% of basic, CCA/MA = Rs. 100, Tax = Rs. 300, PF = Rs.780, TA = Rs. 800).

Total Salary = (Basic + HRA + DA + TA) – (Tax + CM + PF)

1. Perform data flow testing

2. Slice based testing for all variables

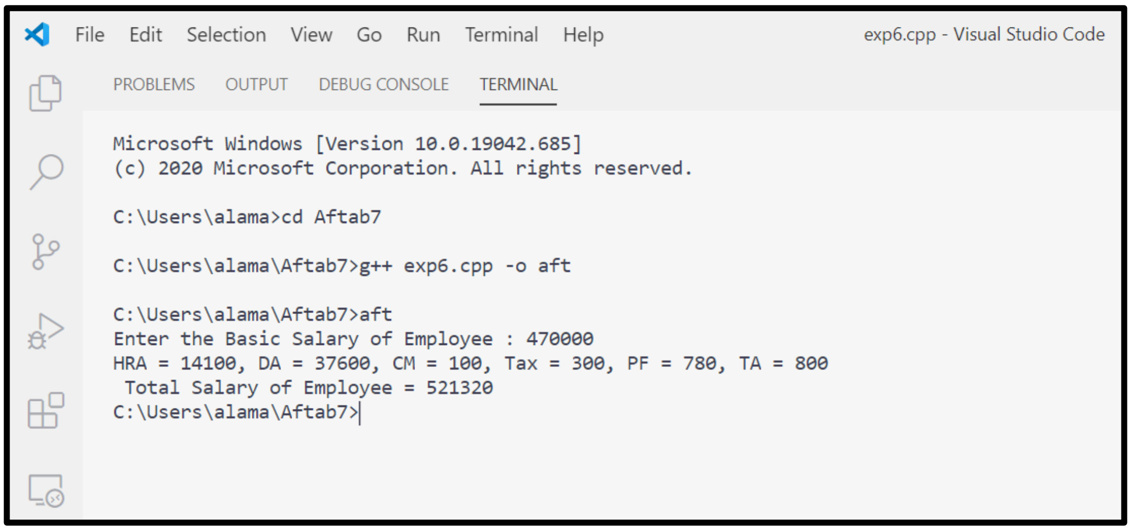
Algorithm :

- Take the Basic Salary of the employee as input from the user.
- Calculate HRA and DA using the basic salary.
- Calculate the Total Salary by combining all the values.
- Print the Total Salary of the employee as calculated on the screen.

Code :

```
#include <iostream>
using namespace std;
int main() {
float basic, HRA, DA, CM, tax, PF, TA, total_salary;
printf("Enter the Basic Salary of Employee : ");
cin >> basic;
HRA = (basic * 3) / 100;
DA = (basic * 8) / 100;
CM = 100;
tax = 300;
PF = 780;
TA = 800;
cout<<"HRA="<<HRA<<"DA="<<DA<<"CM="<<CM<<"Tax="<<tax<<"PF="<<PF<<"TA="<<TA;
total_salary = (basic + HRA + DA + TA) - (tax + CM + PF);
cout << "\n Total Salary of Employee = " << total_salary;
return 0;
}
```

Output Screenshots :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp6.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter the Basic Salary of Employee : 470000
HRA = 14100, DA = 37600, CM = 100, Tax = 300, PF = 780, TA = 800
 Total Salary of Employee = 521320
C:\Users\alama\Aftab7>
```

22/10/2020 **Mohd. Aftab Alam**
02013302717

Data Flow Testing :

Define/Use Table :

S. No.	Variable	Defined at Node	Used at Node
1	basic	2	4,5,6
2	HRA	2,5	12
3	DA	2,6	12
4	CM	2,7	12
5	tax	2,8	7,8
6	PF	2,9	12
7	TA	2,10	12
8	total_salary	2	13

Du-Path Table :

S. No.	Variable	Du-Path (Begin, End)
1	basic	2,4 2,5 2,6
2	HRA	2,12 5,12
3	DA	2,12 6,12
4	CM	2,12 7,12
5	tax	2,12 8,12
6	PF	2,12 9,12
7	TA	2,12 10,12
8	total_salary	2,13

All uses-definition clear :

S. No.	Variable	Du-Path (Begin, End)	Definition clear?
1	basic	2,4	Yes
2	basic	2,4,5,6	Yes
3	HRA	2,5,12	No
4	DA	2,6,12	No
5	CM	2,7,12	No
6	tax	2,8,12	No
7	PF	2,9,12	No
8	TA	2,10,12	No
9	total_salary	2,13	Yes

There is a total of 9 du-paths out of which 6 paths are not defined clearly.

Test case table of all users :

S. No.	Variables			Expected Output	Du-Path
	Basic	HRA	DA		
1	250000	7500	20000	277120.00	2,4
2	350000	10500	28000	388120.00	2,4,5,6
3	450000	13500	36000	499120.00	2,13

Test case table of all definition :

S. No.	Variables			Expected Output	Du-Path
	Basic	HRA	DA		
1	250000	7500	20000	277120.00	2,4
2	350000	10500	28000	388120.00	2,4,5,6
3	450000	13500	36000	499120.00	2,13
4	250000	7500	20000	277120.00	2,4
5	350000	10500	28000	388120.00	2,4,5,6
6	450000	13500	36000	499120.00	2,13

Slice Based Testing :

There is total 8 variables in the program. We can create slices for each of them.

- ***Variable: basic***

S(basic,5) / S(basic,15) = {1-5,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
15.}
```

- ***Variable: HRA***

S(HRA,6) / S(HRA,15) = {1-6,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
15.}
```

- ***Variable: DA***

S(DA,6) / S(DA,15) = {1-6,7,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
6. DA = (basic * 8) / 100;
15.}
```

- ***Variable: CM***

S(CM,8) / S(CM,15) = {1-3,8,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
7. CM = 100;
15.}
```

- ***Variable: tax***

S(tax,8) / S(tax,15) = {1-3,8,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
8. tax = 300;
15.}
```

- ***Variable: PF***

S(PF,10) / S(PF,15) = {1-3,10,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
9. PF = 780;
15.}
```

- ***Variable: TA***

S(TA,11) / S(TA,15) = {1-3,11,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
10.TA = 800;
15.}
```

- ***Variable: total_salary***

S(Total,12) = {1-12,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10.TA = 800;
12.total_salary = (basic + HRA + DA + TA) - (tax + CM + PF);
15.}
```

S(Total,13) / S(Total,15) = {1-13,14,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10. TA = 800;
11.cout<<"HRA="<<HRA<<"DA="<<DA<<"CM="<<CM<<"Tax="<<tax<<"PF="<<PF<<"TA="<<TA;
12. total_salary = (basic + HRA + DA + TA) – (tax + CM + PF);
13. cout << "\n Total Salary of Employee = " << total_salary;
14. return 0;
15. }
```

Test Cases :

S. No.	Slice	Lines Covered	Variables			Expected Output
			Basic	HRA	DA	
1	S(Basic,5) / S(Basic,15)	1-5,15	1000	30	80	No Output
2	S(HRA,6) / S(HRA,15)	1-6,15	3000	90	240	No Output
3	S(DA,7) / S(DA,15)	1-5,7,15	3000	90	240	No Output
4	S(MA,8) / S(MA,15)	1-3,8,15	3000	90	240	No Output
5	S(ITAX,9) / S(ITAX,15)	1-3,9,15	3000	90	240	No Output
6	S(PF,10) / S(PF,15)	1-3,10,15	3000	90	240	No Output
7	S(TA,11) / S(TA,15)	1-3,11,15	3000	90	240	No Output
8	S(Total,12)	1-12,15	3000	90	240	No Output
9	S(Total,13) / S(Total,15)	1-13,14,15	5000	150	400	7530

Aim :

(b) Create a test plan document for any application.

Name of Product: **Web-based Test Management Tool**

Prepared by: **Mohd. Aftab Alam**

Reviewed by: **Ms. Namrata Sukhija** Date: …. /…. /…….

Introduction

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

1.1 Objective

Test Case Tamer is a web-based Test Management tool used to create and store tests as well as the results of running those tests. This tool is a new product written with Ruby on Rails using a MySQL database. The test team is responsible for testing the product and ensuring it meets their needs. The test team is both the customer and the tester in this project.

Phase 1 of the project will deliver TCT (Test Case Tamer) with functionality to create and store manual tests. This will allow the test team to start transferring tests over to the new system. Must have functionality is considered more important than the delivery date in this project.

1.2 Team Members

Resource Name	Role
	DEVELOPER
	TESTER
	DESIGNER

2. Scope

The initial phase will include all 'must have' requirements. These and any other requirements that get included must all be tested. At the end of Phase 1, a tester must be able to:

i. Create a manual test with as many steps as necessary
ii. Save it
iii. Retrieve it and have the ability to view it when running the test
iv. Enter results and appropriate comments
v. View results

As the team works with the product, they will define the needs for the second phase. Load testing will not be considered part of this project since the user base is known and not an issue.

Rewriting, moving or porting existing test cases from the existing Word documents is not considered part of this project.

3. Assumptions and Risks

3.1 Assumptions

This section lists assumptions that are made specific to this project. Delivery of the product is in format that the test team can check it into CVS.

3.2 Risks

The following risks have been identified and the appropriate action identified to mitigate their impact on the project. The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone or event would cause the risk to become an issue to be dealt with.

#	Risk	Impact	Trigger	Mitigation Plan
1	Scope Creep - as testers become more familiar with the tool, they will want more functionality	High	Delays in implementation date	Each iteration, functionality will be closely monitored. Priorities will be set and discussed by stakeholders
2	Changes to the functionality may negate the tests already written and we may lose test cases already written	High-to schedule and quality	Loss of all test cases	Export data prior to any upgrade, massage as necessary and re-import after upgrade
3	Weekly delivery is not possible because the developer works off site	Medium	Product did not get delivered on schedule	

4. Test Approach

The project is using an agile approach, with weekly iterations. At the end of each week the requirements identified for that iteration will be delivered to the team and will be tested. Exploratory testing will play a large part of the testing as the team has never used this type of tool and will be learning as they go. Tests for planned functionality will be created and added to TCT as we get iterations of the product.

4.1 Test Automation

Automated unit tests are part of the development process, but no automated functional tests are planned at this time.

5. Test Environment

A new server is required for the web server, the application and the database.

6. Milestones / Deliverables

6.1 Test Schedule

The initial test schedule follows…

Task Name	Start	Finish	Effort	Comments
Test Planning				
Review Requirements documents			2 d	
Create initial test estimates			1 d	
Staff and train new test resources				
First deploy to QA test environment				
Functional testing – Iteration 1				
Iteration 2 deploy to QA test environment				
Functional testing – Iteration 2				
System testing				
Regression testing				
UAT				
Resolution of final defects and final build testing				
Deploy to Staging environment				
Performance testing				
Release to Production				

6.2 Deliverables

Deliverable	For	Date / Milestone
Test Plan	Project Manager; QA Director; Test Team	
Traceability Matrix	Project Manager; QA Director	
Test Results	Project Manager	
Test Status Report	QA Manager, QA Director	
Metrics	All Team Members	

22/10/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 7

Aim :

(a) Write a program in C/C++ for classification of Triangle on the basis of angle. Its input is triple of positive integers (say a, b, c) and values may be from the interval [1, 100]. The output may have one of the following: Right Angled, Acute Angled, Obtuse Angled or Not a Triangle.

Perform Slice-based testing and generate test cases.

Algorithm :

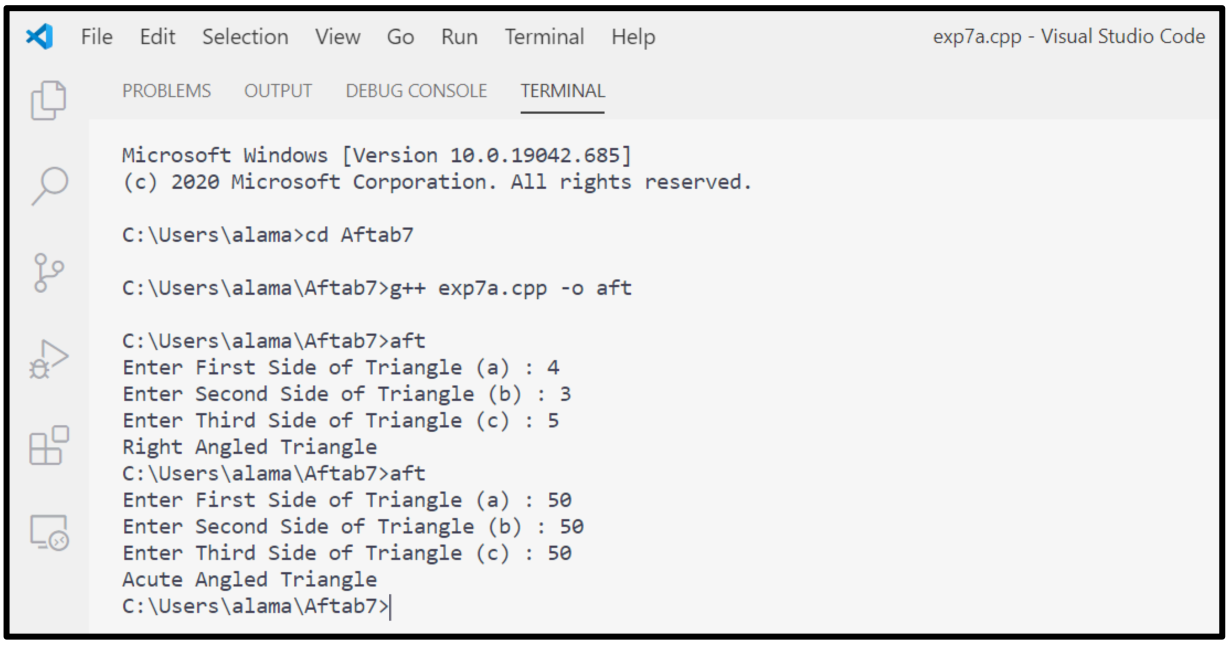
- Take three inputs from the user as the sides of the triangle.
- Check whether they lie in the given interval.
- If the condition is true then check if it is a Triangle or not.
- If the condition is true then put valid = 1, otherwise, put valid = -1.
- If valid = 1 then calculate the value of a1, a2, and a3 as follows:
    - **a1 = (a2 + b2) / c2**
    - **a2 = (b2 + c2) / a2**
    - **a3 = (c2 + a2) / b2**
- Check if a1, a2, or a3 are less than 1.
- If the condition is true then it is an Obtuse Angled Triangle.
- Else if a1, a2 or a3 are equal to 1.
- If the condition is true then it is a Right-Angled Triangle.
- If the condition is false then it is an Acute Angled Triangle.
- Else if valid = -1, then it is an Invalid Triangle.
- Else the input values are Out of Range.

Code :

```cpp
#include <iostream>
using namespace std;

int main()
{
    float a, b, c;
    float a1, a2, a3;
    int valid = 0;
    cout << "Enter First Side of Triangle (a) : ";
    cin >> a;
    cout << "Enter Second Side of Triangle (b) : ";
    cin >> b;
    cout << "Enter Third Side of Triangle (c) : ";
    cin >> c;
    if (a > 0 && a <= 100 && b > 0 && b <= 100 && c > 0 && c <= 100)
    {
        if ((a + b) > c && (b + c) > a && (c + a) > b)
            valid = 1;
        else
            valid = -1;
    }
    if (valid == 1)
    {
        a1 = (a * a + b * b) / (c * c);
        a2 = (b * b + c * c) / (a * a);
        a3 = (c * c + a * a) / (b * b);
        if (a1 < 1 || a2 < 1 || a3 < 1)
            cout << "Obtuse Angled Triangle";
        else if (a1 == 1 || a2 == 1 || a3 == 1)
            cout << "Right Angled Triangle";
        else
            cout << "Acute Angled Triangle";
    }
    else if (valid == -1)
        cout << "Invalid Triangle";
    else
        cout << "Out of Range";
    return 0;
}
```

Output Screenshots :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp7a.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter First Side of Triangle (a) : 4
Enter Second Side of Triangle (b) : 3
Enter Third Side of Triangle (c) : 5
Right Angled Triangle
C:\Users\alama\Aftab7>aft
Enter First Side of Triangle (a) : 50
Enter Second Side of Triangle (b) : 50
Enter Third Side of Triangle (c) : 50
Acute Angled Triangle
C:\Users\alama\Aftab7>
```

22/10/2020 **Mohd. Aftab Alam**
02013302717

Slice Based Testing :

There is total 7 variables in the program. We can create slices for each of them.

1. S(a,8)/S(a,42) = {1-8,42}
2. S(b,10) / S(b,42) = {1-6,9,10,42}
3. S(c,12) / S(c,42) = {1-6,11,12,42}
4. S(a1,22) / S(a1,42) = {1-16,20-22,34,42}
5. S(a1,26) = {1-16,20-22,25-27,34,41,42}
6. S(a1,29) = {1-16,20-22,25,27-31,33,34,41,42}
7. S(a1,32) = {1-16,20-22,25,27,28,30-34,41,42}
8. S(a2,23) /S(a2,42) = {1-16,20,21,23,34,42}
9. S(a2,26) = {1-16,20,21,23,25-27,34,41,42}
10. S(a2,29) = {1-16,20,21,23,25,27-31,33,34,41,42}
11. S(a2,32) = {1-16,20,21,23,25,27,28,30-34,41,42}
12. S(a3,24) / S(a3,42) = {1-16,20,21,24,34,42}
13. S(a3,26) = {1-16,20,21,24-27,34,41,42}
14. S(a3,29) = {1-16,20,21,24,25,27-31,33,34,41,42}
15. S(a3,32) = {1-16,20,21,24,25,27,28,30-34,41,42}
16. S(valid,5) = {1-5,42}
17. S(valid,15) = {1-16,20,42}
18. S(valid,18) = {1-14,16-20,42}
19. S(valid,36) = {1-14,16-20,21,34-38,40-42}
20. S(valid,39) = {1-13,20,21,34,35,37-42}
21. S(valid,42) = {1-14,16-20,42}

Test Cases :

S. No.	Slice	Lines Covered	Variables			Expected Output
			a	b	c	
1	S(a,8)/S(a,42)	1-8,42	20			No Output
2	S(b,10)/S(b,42)	1-6,9,10,42		20		No Output
3	S(c,12)/S(c,42)	1-6,11,12,42			20	No Output
4	S(a1,22)/S(a1,42)	1-16,20-22,34,42	30	20	40	No Output
5	S(a1,26)	1-16,20-22,25-37,34,41,42	30	20	40	Obtuse Angled
6	S(a1,29)	1-16,20-22,25,27-31, 33,34,41,42	30	40	50	Right Angled
7	S(a1,32)	1-16,20-22,25,27,30-34,41,42	50	60	40	Acute Angled
8	S(a2,23)/S(a2,42)	1-16,20,21,23,34,42	30	20	40	No Output
9	S(a2,26)	1-16,20,21,23,25-27, 34,41,42	40	30	20	Obtuse Angled
10	S(a2,29)	1-16,20,21,23,25,27-31, 33,34,41,42	50	40	30	Right Angled
11	S(a2,32)	1-16,20,21,23, 25,27,28,30-34,41,42	40	50	60	Acute Angled
12	S(a3,24)/S(a3,42)	1-16,20,21,24,34,42	30	20	40	No Output
13	S(a3,26)	1-16,20,21,24-27,34,41,42	20	40	30	Obtuse Angled
14	S(a3,29)	1-16,20,21,24,25,27-31, 33,34,41,42	40	50	30	Right Angled
15	S(a3,32)	1-16,20,21,24,25,27,28, 30-34,41,42	50	40	60	Acute Angled
16	S(valid,5)	1-5,42				No Output
17	S(valid,15)	1-16,20,42	20	40	30	No Output
18	S(valid,18)	1-14,18-20,42	30	10	15	No Output
19	S(valid,36)	1-14,16-20,21,34-38,40-42	30	10	15	Invalid Triangle
20	S(valid,39)	1-13,20,21,34,35,37-42	102	-1	6	Out of Range
21	S(valid,42)	1-14,16-20,42	30	10	15	No Output

22/10/2020 **Mohd. Aftab Alam**
02013302717

Aim :

(b) Write a Program in C/C++ to compute total salary of an employee when his basic salary is given.

(Given: HRA = 3% of basic, DA = 8% of basic, CCA/MA = Rs. 100, Tax = Rs. 300, PF = Rs.780, TA = Rs. 800). Perform Slice based testing for all variables

Total Salary = (Basic + HRA + DA + TA) – (Tax + CM + PF)

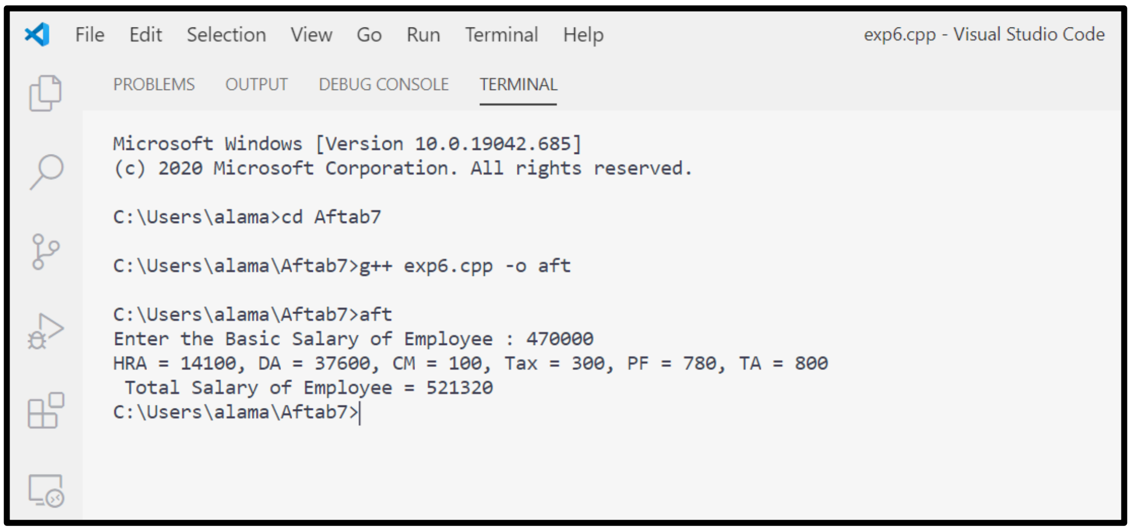
Algorithm :

- Take the Basic Salary of the employee as input from the user.
- Calculate HRA and DA using the basic salary.
- Calculate the Total Salary by combining all the values.
- Print the Total Salary of the employee as calculated on the screen.

Code :

```
#include <iostream>
using namespace std;
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf(“Enter the Basic Salary of Employee : “);
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10. TA = 800;
11.cout<<”HRA=”<<HRA<<”DA=”<<DA<<”CM=”<<CM<<”Tax=”<<tax<<”PF=”<<PF<<”TA=”<<TA;
12. total_salary = (basic + HRA + DA + TA) – (tax + CM + PF);
13. cout << “\n Total Salary of Employee = “ << total_salary;
14. return 0;
15. }
```

Output Screenshots :


File Edit Selection View Go Run Terminal Help
exp6.cpp - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp6.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter the Basic Salary of Employee : 470000
HRA = 14100, DA = 37600, CM = 100, Tax = 300, PF = 780, TA = 800
 Total Salary of Employee = 521320
C:\Users\alama\Aftab7>
```

Slice Based Testing :

There is total 8 variables in the program. We can create slices for each of them.

- ***Variable: basic***

S(basic,5) / S(basic,15) = {1-5,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
15.}
```

- ***Variable: HRA***

S(HRA,6) / S(HRA,15) = {1-6,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
15.}
```

- ***Variable: DA***

S(DA,6) / S(DA,15) = {1-6,7,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
6. DA = (basic * 8) / 100;
15.}
```

- ***Variable: CM***

S(CM,8) / S(CM,15) = {1-3,8,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
7. CM = 100;
15.}
```

- ***Variable: tax***

S(tax,8) / S(tax,15) = {1-3,8,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
8. tax = 300;
15.}
```

- ***Variable: PF***

S(PF,10) / S(PF,15) = {1-3,10,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
9. PF = 780;
15.}
```

- ***Variable: TA***

S(TA,11) / S(TA,15) = {1-3,11,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
10.TA = 800;
15.}
```

- ***Variable: total_salary***

S(Total,12) = {1-12,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10.TA = 800;
12.total_salary = (basic + HRA + DA + TA) - (tax + CM + PF);
15.}
```

S(Total,13) / S(Total,15) = {1-13,14,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf(“Enter the Basic Salary of Employee : “);
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10. TA = 800;
11.cout<<”HRA=”<<HRA<<”DA=”<<DA<<”CM=”<<CM<<”Tax=”<<tax<<”PF=”<<PF<<”TA=”<<TA;
12. total_salary = (basic + HRA + DA + TA) – (tax + CM + PF);
13. cout << “\n Total Salary of Employee = “ << total_salary;
14. return 0;
15. }
```

22/10/2020 **Mohd. Aftab Alam**
02013302717

Test Cases :

S. No.	**Slice**	**Lines Covered**	**Variables**			**Expected Output**
			Basic	**HRA**	**DA**	
1	S(Basic,5) / S(Basic,15)	1-5,15	1000	30	80	No Output
2	S(HRA,6) / S(HRA,15)	1-6,15	3000	90	240	No Output
3	S(DA,7) / S(DA,15)	1-5,7,15	3000	90	240	No Output
4	S(MA,8) / S(MA,15)	1-3,8,15	3000	90	240	No Output
5	S(ITAX,9) / S(ITAX,15)	1-3,9,15	3000	90	240	No Output
6	S(PF,10) / S(PF,15)	1-3,10,15	3000	90	240	No Output
7	S(TA,11) / S(TA,15)	1-3,11,15	3000	90	240	No Output
8	S(Total,12)	1-12,15	3000	90	240	No Output
9	S(Total,13) / S(Total,15)	1-13,14,15	5000	150	400	7530

22/10/2020 **Mohd. Aftab Alam**
02013302717

Aim :

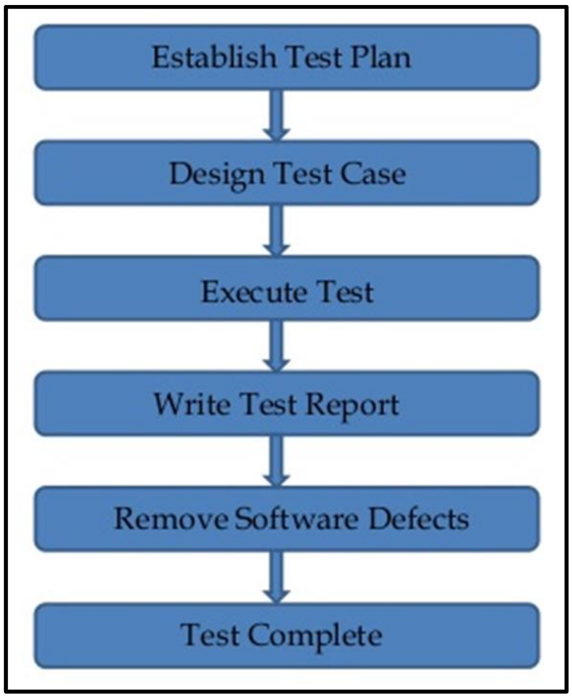
(c) To study a Testing Tool – **WinRunner**

1. Introduction

If you have ever tested software manually, you are aware of its drawbacks. Manual testing is time-consuming and tedious, requiring a heavy investment in human resources. Worst of all, time constraints often make it impossible to manually test every feature thoroughly before the software is released. This leaves you wondering whether serious bugs have gone undetected.

Automated testing with WinRunner addresses these problems by dramatically speeding up the testing process. You can create test scripts that check all aspects of your application, and then run these tests on each new build. As WinRunner runs tests, it simulates a human user by moving the mouse cursor over the application, clicking Graphical User Interface (GUI) objects, and entering keyboard input-but WinRunner does this faster than any human user.

2. Process Flow for WinRunner



3. Features

- Functional Regression Testing Tool
- Windows Platform Dependent
- Only for Graphical User Interface (GUI) based Application
- Based on Object Oriented Technology (OOT) concept
- Only for Static content
- Developed by Mercury Interactive
- Functionality testing tool
- WinRunner run on Windows only
- XRunner run only in UNIX and Linux
- Tool developed in C on VC++ environment
- To automate our manual test Win runner used TSL (Test Script language like C)
- Record/Playback Tool

4. Add-ins

WinRunner includes the following Add-ins:

- Web Test
- Visual Basic
- ActiveX
- Power Builder

5. Running the Test

WinRunner provides three modes for running test:

- Use Verify mode when running a test to check the behaviour of our application and when we want to save the test result.
- Use Debug Mode, when you want to check that the test script runs smoothly without errors in syntax. The debug mode will not give the test result.
- Use Update mode, when you want to create new expected results for a GUI check point or bitmap check point.

6. WinRunner Testing Process

- Create GUI Map File: By creating GUI Map file the WinRunner can identify the GUI objects in the application going to be tested.
- Create Test Scripts: This process involves recording, programming or both. During the process of recording tests, insert checkpoints where the response of the application needs to be tested.
- Debug Test: Run the tests in Debug mode to make sure whether they run smoothly.
- Run Tests: Run tests in Verify mode to test the application.
- View Results: This determines the success or failure of the tests.
- Report Defects: If a particular test run fails due to the defect in the application being tested, defects can be directly reported through the Test Results window.

7. Facts about WinRunner

- WinRunner doesn’t support web application.
- WinRunner only supports IE and Netscape Navigator.
- The default time setting of WinRunner is 10000ms.
- The logical name of true and false in WinRunner is 1 and 0.
- WinRunner doesn’t provide the snapshot of output.
- WinRunner has 3 kinds of checkpoints, 2 kinds of Recordings and 3 kinds of Exception Handling mechanisms.
- In WinRunner GUI, Spy is available.
- WinRunner uses TSL script (Test Script Language).
- WinRunner only works on 32-bit machine.

EXPERIMENT 8

Aim : (a) To study a Test Management Tool – **QA Complete**

Introduction

Take the guess work out of the software delivery lifecycle. Provide your QA and development teams with the power to collaborate, track project progress, and report on requirements, test cases, and defects.

QA Complete allows you to take a strategic approach to testing by prioritizing key test functions, accounting for risk, planning for coverage, and controlling test execution. Employing effective test case management helps you ensure you're running the right tests, and thus avoid releasing an application that is not customer-ready.

Key Features

1. Test Case Management

The ability to organize, plan, and analyse your testing efforts across the lifecycle is critical to your success or failure whether you use manual or automated test cases today. As projects cope with fewer development resources, higher quality expectations, and shorter development timelines, any serious development effort needs better test case management. QA Complete delivers:

- Manage manual test cases and link them back to the original requirements, thereby ensuring a requirement has been met. Evaluate the test run history of those automated tests right from QA Complete.
- Employ re-usable manual test libraries to quickly create new test scenarios.
- Graphically report automated test runs with plug-ins for many leading automated testing tools, including Test Complete and HP Quick Test Pro (QTP).
- Add, print, edit, or email test cases with a single click.

2. Test Automation Tool Integration

QA Complete supports many automated testing tools, including Automated QA Test Complete and HP Quick Test Pro. Integration with test automation tools allows you review the run history of any automated test on any machine, so if you have a test lab with multiple machines running automated tests, you can compare machine run history. Since you can co-ordinate both manual and automated tests, you have better test information to make release decisions.

By integrating automated testing into QA Complete, you can:

- Launch the tests from within your automated tool and automatically report the run information to QA Complete for analysis of runs over time.
- Trend results using graphical dashboards and schedule tests to run unattended.

3. Bi-Directional Traceability

The goal of traceability is to ensure "adequate" test coverage for each software requirement. It is important to maintain traceability both forwards and backwards, from requirement to test case and from test case to requirement. This ensures that design specifications are appropriately verified and that requirements are appropriately validated, ultimately reducing software defects and – this is the ultimate goal, after all – improving code quality.

- Link together requirements, test cases, and reported defects.
- Drag and drop functionality to link test cases or defects to a requirement.
- With one click, see a traceability report showing all linkages to a particular requirement.

4. Requirements Management

QA Complete helps you manage requirements regardless of your team's development methodology. It lets you define requirements for each release and track the release for which each requirement is scheduled. Govern workflow and state transitions. Using workflow, you can force design reviews, approvals, or test reviews.

- Easily set rules for status transitions.
- Automatically assign tasks to team members based on requirement status; receive email alerts when requirements change.
- Attach documents to any requirement (such as detail designs, specifications, and prototypes), and track versions of those documents.
- Add notes to the requirement, allowing the team to discuss requirement changes and other important issues.
- View an audit history of any requirement, showing who made a change, the date and time of the change, and the before-and-after values.
- Rely on an extensive set of standard and ad-hoc dashboards and reports (requirements missing test cases, etc.).

5. Defect and Issue Tracking

QA Complete allows you to track status and resolution progress of defects and issues for each release. Instead of spending your time entering data, the software automatically generates a defect identifier on failed test cases.

Integration with Atlassian JIRA, Bugzilla, and other web-based defect tracking tools allow you to blend QA Complete features with the defect tracking tools your organization already uses.

- Coordinate QA and development teams to coordinate activities as bugs are found. QA Complete has a full featured defect tracking component.
- If your team already owns a bug tracking system (like JIRA, Bugzilla, Microsoft TFS, etc.), you can create defects inside of QA Complete and have those automatically synchronized with your bug tracking system.
- Source code integration lets you associate source code with fixed defects. By doing this, you can quickly discover which code modules are the buggiest, allowing your team to put effort into raising the quality of that code.
- Defect reports and dashboards show defects by severity, priority, or other criteria.

6. Seamless Integration with other ALM Tools

If your team has already sunk costs into an existing requirements or defect management tools, it may be difficult to convince your team to switch. With QA Complete, you don't have to switch. QA Complete can seamlessly integrate with many defects and ALM tools including Microsoft Team Foundation Server (TFS), HP Quality Centre, IBM Rational Doors, IBM Rational Requisite Pro, Rational Team Concert, Atlassian JIRA, Rally, Version One, Bugzilla, and Accept 360.

7. Seamless Integration with Source Control Systems

Associating defects and requirements with source code provides great traceability, allowing you to quickly discover troublesome code and requirements that required the most re-work. Using the QA Complete SCM integrators, you can associate defects or requirements when checking in source code.

8. Extensive Dashboards and Reporting for QA Activities

QA Complete provides an array of analysis tools. Status screens, dashboards, and reports help you stay on track, better plan your next release, and answer the most pressing questions about your software development projects. For example:

- Are you progressing properly in your test cycle? The Test Case Trending dashboard shows how many test cases are still awaiting run, how many passed, and how many have failed, day-by-day.
- Which tester has the most assigned defects? View a Defect Assignee report that shows defects by assignee. Click the assignee graph to get further details, print, export to Word, Excel or PDF, or email the results.
- Review team and project variances, allowing you to you learn from each project as to improve future estimates.

9. Easy Data Entry

QA Complete makes it easy for you to automatically generate a defect from a failed test case. With one-click cloning of records, you can pass along to your development team all the relevant steps to reproduce the problem, along with expected and actual results, without having to re-enter anything.

- Automatically create a link between the defect and the test case.
- Existing test cases linked to a requirement automatically link the defect to the requirement as well for full traceability.

10. Flexible Licensing Model

Our QA Complete SaaS offering allows you to focus on your core business activities while keeping costs down. Our online option means you don’t have to mess with server configuration, hardware support, or other data centre annoyances. Sign up to get up and running immediately and leave the hosting to us.

- On-demand, available anytime, from anywhere.
- Sign up and turn it on with no implementation effort.
- Safe and secure Data Centre.
- For those who would prefer to host the software, or are unable to use a SaaS model, we also offer an on-premise option. Both are backed by our outstanding and friendly support staff.

11. Web Based User Interface

QA Complete has a Web interface. Nothing needs to be installed on your hard drive.

- Users with an Internet connection may access QA Complete from home or anywhere in the world.
- Distributed and offshore teams can easily share QA artifacts with one another and local teams.
- Supports all major browsers, including Internet Explorer, Fire Fox, Safari, and Google Chrome.

Aim : (b) Automate the Test Cases using Test Automation Tool **– QA Complete**

Introduction

QA Complete is a test management tool that can be used for both manual and automated testing. It is a tool with powerful test management capabilities that allow us to track all aspects of software quality in an efficient manner. QA Complete supports all aspects of test process and ensures the delivery of high-quality software.

Benefits

1. QA Complete can be integrated with any number of tools
2. Customizable as per the tester's needs
3. Requirements and tests can be traced to defect effectively
4. Reports and dashboards are the key features of QA Complete

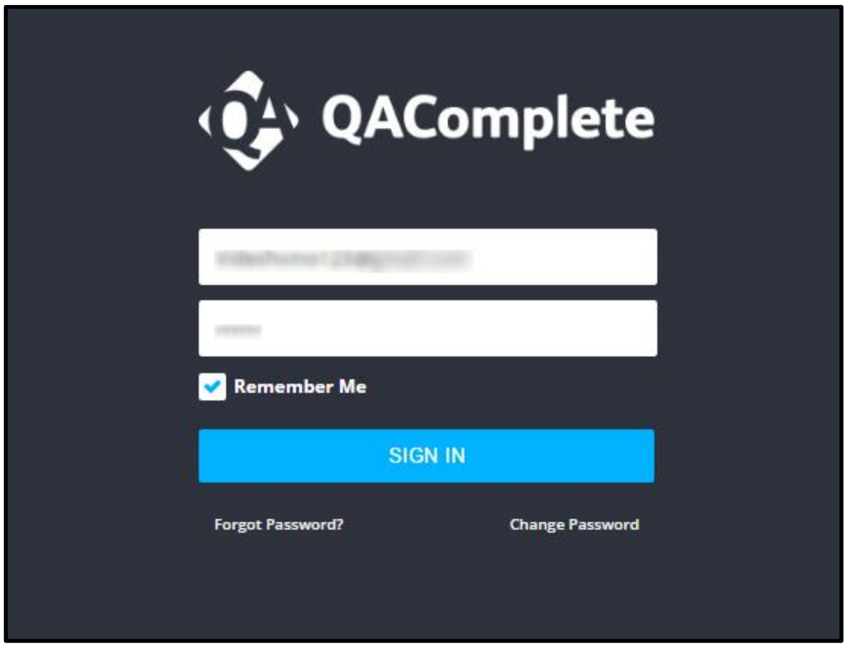
Features

1. Test case management – simple test case structuring also allows for focused metrics and clear status report
2. Test environment management – various environments are linked to individual test cases for effective test coverage across different platforms, operating systems, and devices
3. Defect and issue management – mainly tracks the resolution process of bugs for each release and automatically creates bugs when test cases fail
4. Test automation integration – can be integrated with various automation tools, and it allows us to track and report overall (manual and automated) test management efforts at a single place
5. Bug tracker integration – can be integrated with various bug tracking tools
6. Shared documents – stores all test documents in one central location to improve collaboration

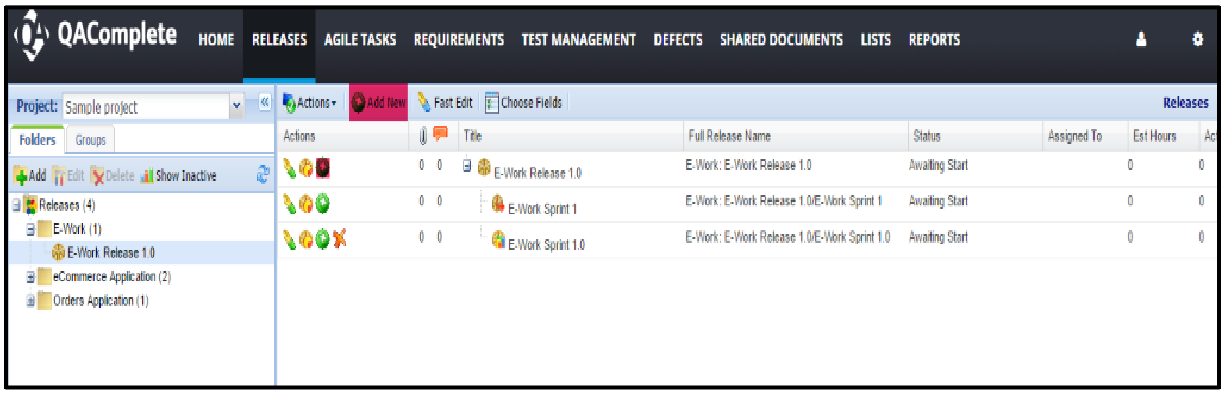
Steps to Setup and Work on QA Complete

To manage and produce the right test deliverables, let us assume an e-work sight login page needs to be tested manually. The following steps and screenshots will explain how we can manage the test deliverables using QA Complete test management tool.

Step 1: Log into QA Complete Tool

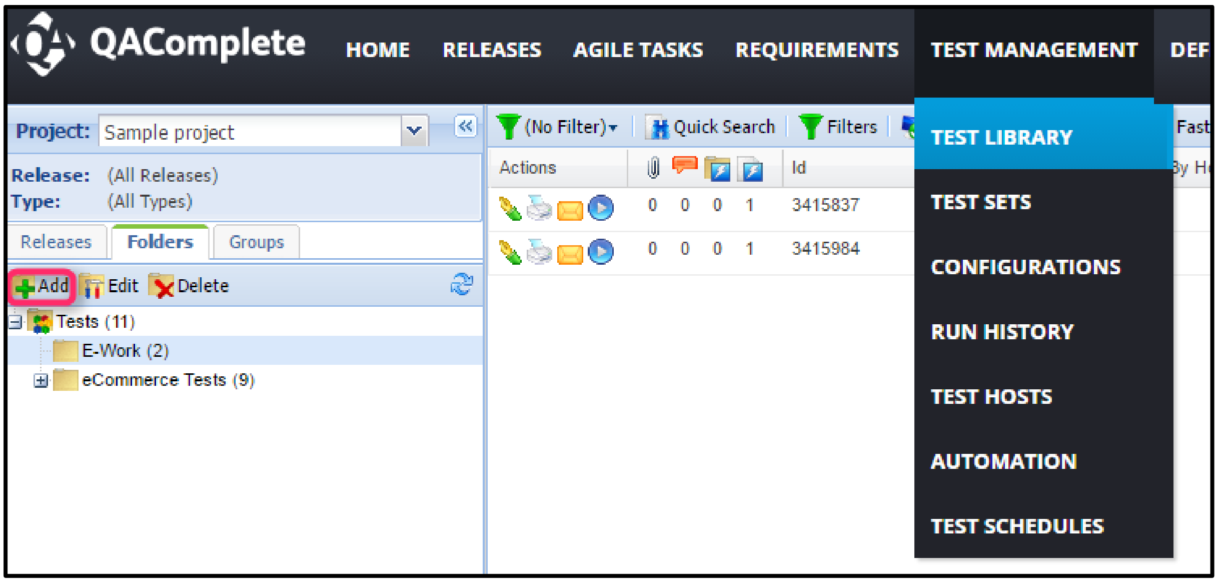


Step 2: Create a new release as e-work under the release tab by clicking the add new icon and click the add new item to add an iteration/build.



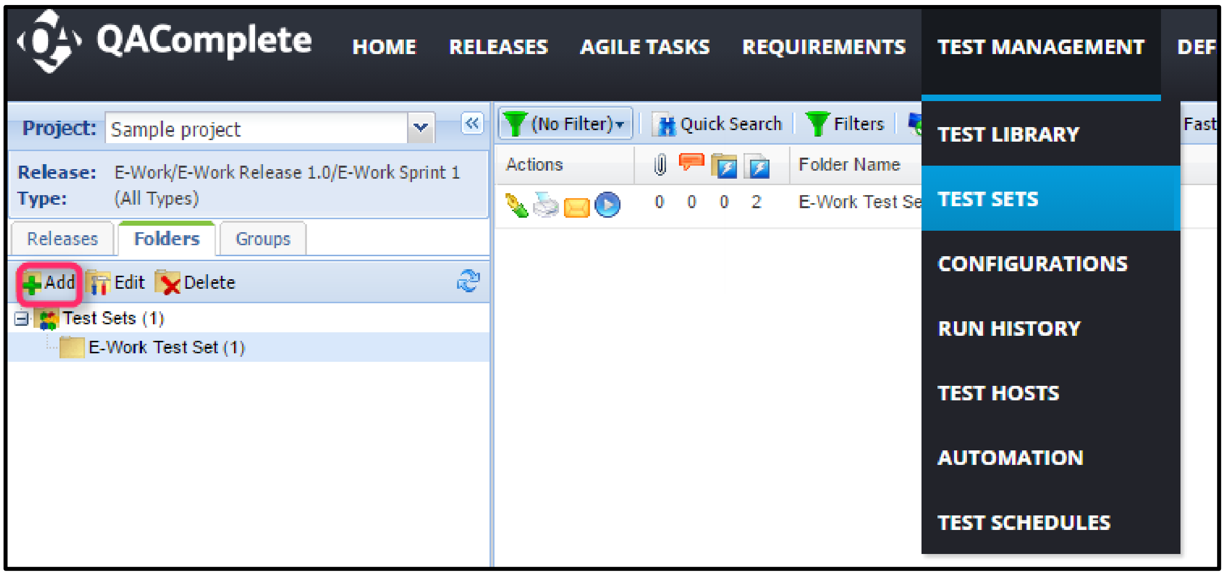
Actions			Title	Full Release Name	Status	Assigned To	Est Hours	Ac
	0	0	E-Work Release 1.0	E-Work: E-Work Release 1.0	Awaiting Start		0	0
	0	0	E-Work Sprint 1	E-Work: E-Work Release 1.0/E-Work Sprint 1	Awaiting Start		0	0
	0	0	E-Work Sprint 1.0	E-Work: E-Work Release 1.0/E-Work Sprint 1.0	Awaiting Start		0	0

Step 3: Navigate to the test management tab -> test library -> add new folder.

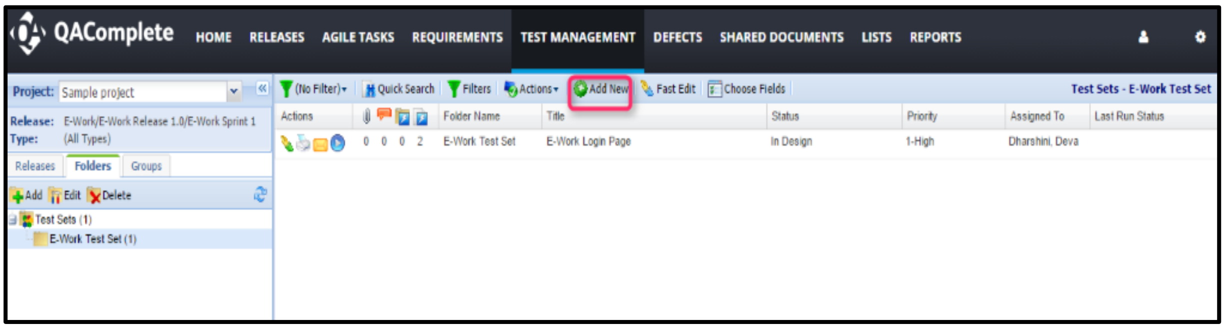


Actions					Id
	0	0	0	1	3415837
	0	0	0	1	3415984

Step 4: Navigate to test management tab -> test sets -> create a folder (add the '+' @ left panel) -> create a new test set using the add new icon -> after entering the details click the submit button -> navigate to the tests tab and design the steps accordingly.



Actions					Folder Name
	0	0	0	2	E-Work Test Se

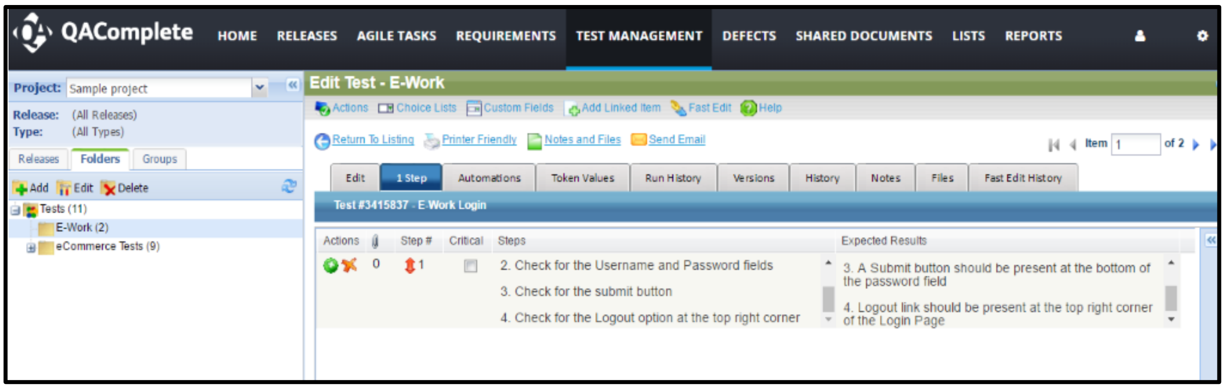


Project: Sample project
Release: E-Work/E-Work Release 1.0/E-Work Sprint 1
Type: (All Types)

Test Sets (1)
E-Work Test Set (1)

Test Sets - E-Work Test Set

Actions					Folder Name	Title	Status	Priority	Assigned To	Last Run Status
	0	0	0	2	E-Work Test Set	E-Work Login Page	In Design	1-High	Dharshini, Deva	



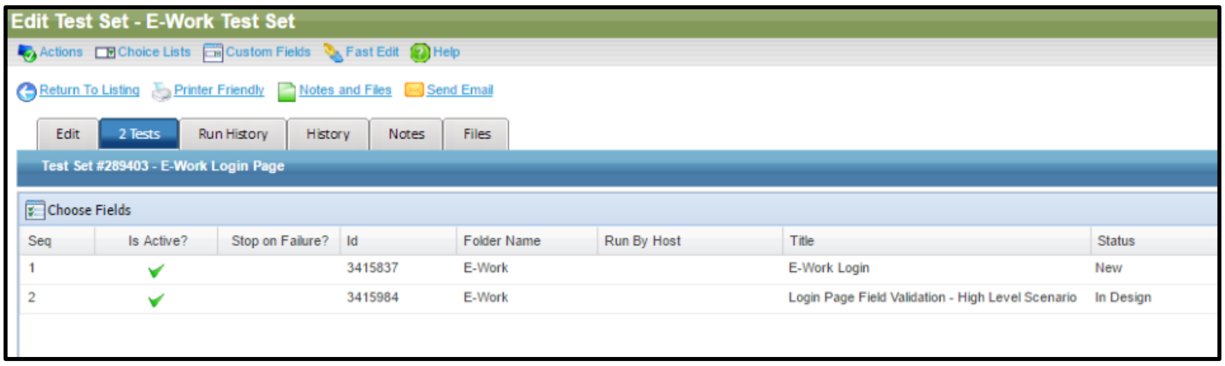
Project: Sample project
Release: (All Releases)
Type: (All Types)

Tests (11)
E-Work (2)
eCommerce Tests (9)

Edit Test - E-Work

Test #3415837 - E-Work Login

Actions		Step #	Critical	Steps	Expected Results
	0	1		2. Check for the Username and Password fields 3. Check for the submit button 4. Check for the Logout option at the top right corner	3. A Submit button should be present at the bottom of the password field 4. Logout link should be present at the top right corner of the Login Page

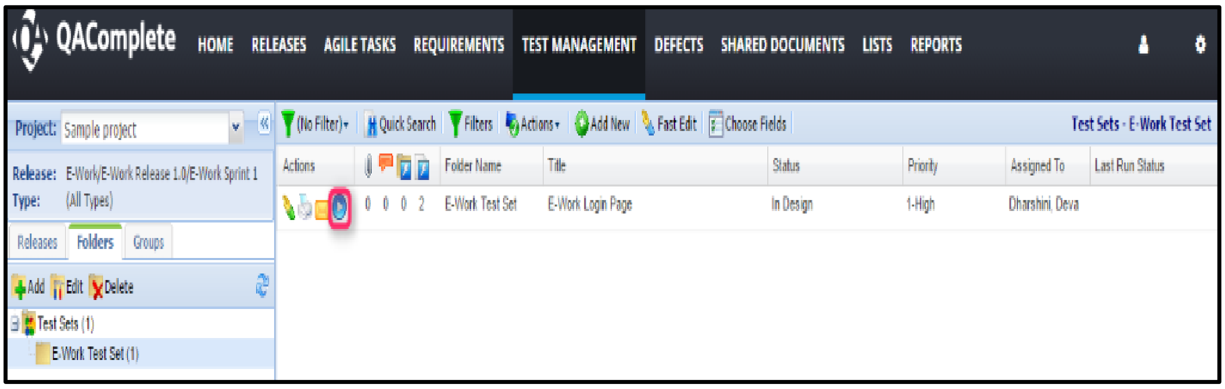


Edit Test Set - E-Work Test Set

Test Set #289403 - E-Work Login Page

Seq	Is Active?	Stop on Failure?	Id	Folder Name	Run By Host	Title	Status
1	✓		3415837	E-Work		E-Work Login	New
2	✓		3415984	E-Work		Login Page Field Validation - High Level Scenario	In Design

Step 5: To run the test sets -> test management tool -> test sets -> click the run icon.



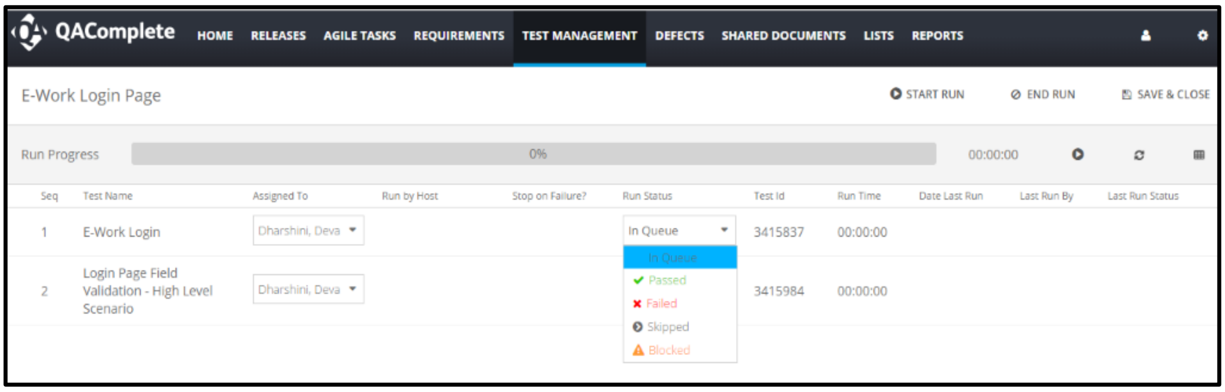
Project: Sample project
Release: E-Work/E-Work Release 1.0/E-Work Sprint 1
Type: (All Types)

Test Sets (1)
E-Work Test Set (1)

Test Sets - E-Work Test Set

Actions					Folder Name	Title	Status	Priority	Assigned To	Last Run Status
	0	0	0	2	E-Work Test Set	E-Work Login Page	In Design	1-High	Dharshini, Deva	

Step 6: Click the start run at the top right corner. Based on the working functionality, select the run status and click the End-Run option finally.

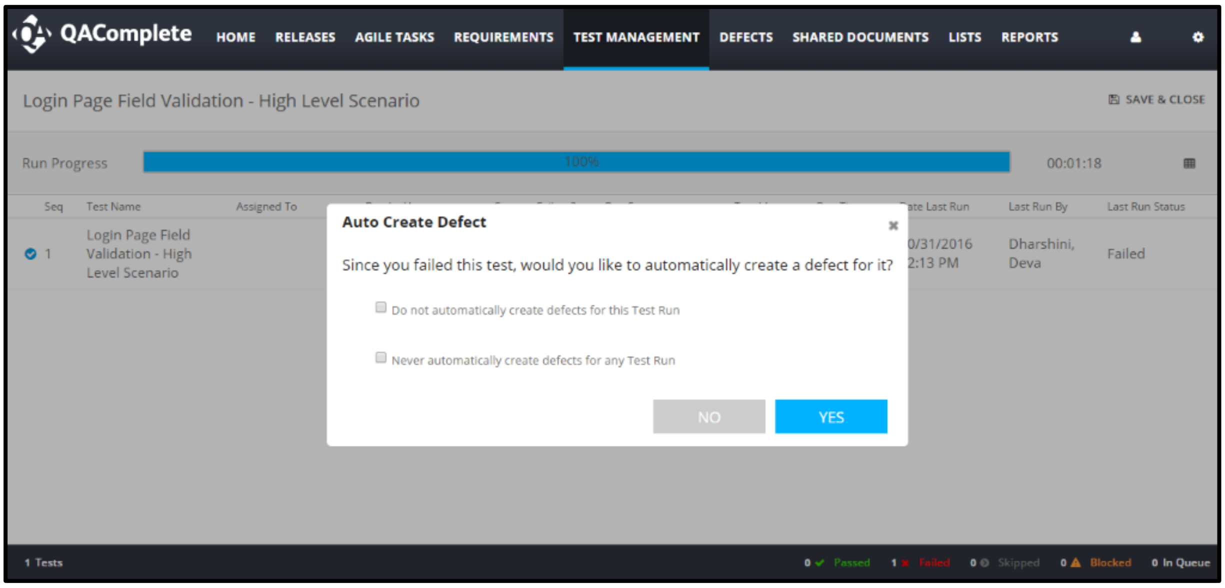


E-Work Login Page

Run Progress 0% 00:00:00

Seq	Test Name	Assigned To	Run by Host	Stop on Failure?	Run Status	Test Id	Run Time	Date Last Run	Last Run By	Last Run Status
1	E-Work Login	Dharshini, Deva			In Queue	3415837	00:00:00			
2	Login Page Field Validation - High Level Scenario	Dharshini, Deva				3415984	00:00:00			

Step 7: If any of the steps fail in a test set during the run, it prompts to create a defect automatically.



Login Page Field Validation - High Level Scenario

Run Progress 100% 00:01:18

Seq	Test Name	Assigned To	Date Last Run	Last Run By	Last Run Status
1	Login Page Field Validation - High Level Scenario		0/31/2016 2:13 PM	Dharshini, Deva	Failed

Auto Create Defect

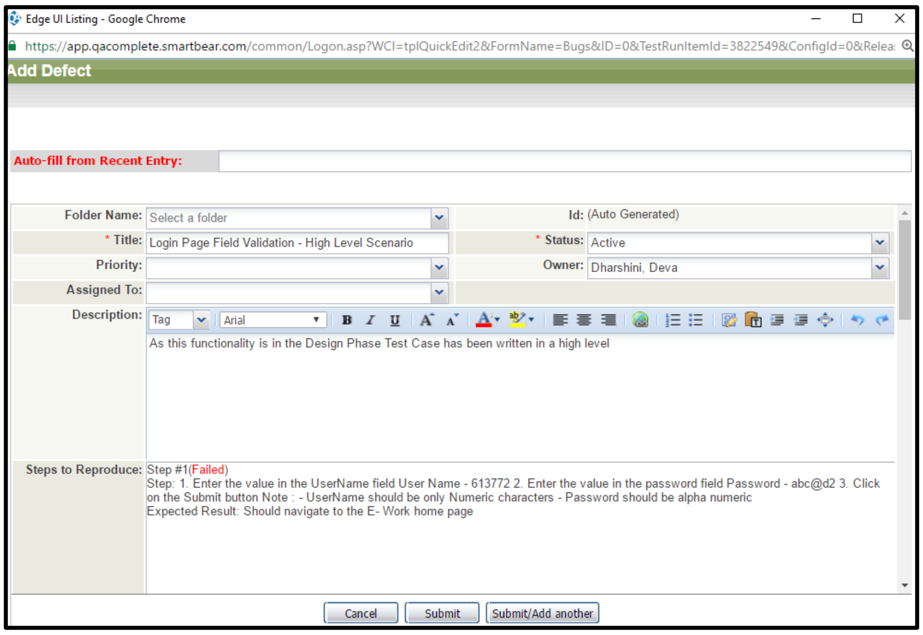
Since you failed this test, would you like to automatically create a defect for it?

☐ Do not automatically create defects for this Test Run

☐ Never automatically create defects for any Test Run

1 Tests 0 ✓ Passed 1 ✗ Failed 0 Skipped 0 Blocked 0 In Queue

Step 8: When the yes option is selected, a defect is created automatically.



https://app.qacomplete.smartbear.com/common/Logon.asp?WCI=tplQuickEdit2&FormName=Bugs&ID=0&TestRunItemId=3822549&ConfigId=0&Relea

Add Defect

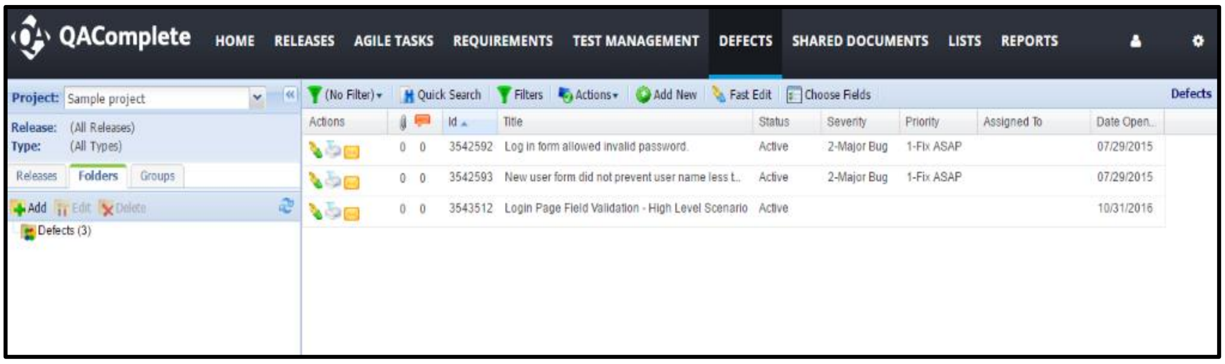
Auto-fill from Recent Entry:

Folder Name:	Select a folder	Id:	(Auto Generated)
* Title:	Login Page Field Validation - High Level Scenario	* Status:	Active
Priority:		Owner:	Dharshini, Deva
Assigned To:			

Description: As this functionality is in the Design Phase Test Case has been written in a high level

Steps to Reproduce: Step #1(Failed)
Step: 1. Enter the value in the UserName field User Name - 613772 2. Enter the value in the password field Password - abc@d2 3. Click on the Submit button Note : - UserName should be only Numeric characters - Password should be alpha numeric
Expected Result: Should navigate to the E- Work home page

Step 9: Navigate to the defects tab and view the automatically created bug(s).



Actions			Id	Title	Status	Severity	Priority	Assigned To	Date Open..
	0	0	3542592	Log in form allowed invalid password.	Active	2-Major Bug	1-Fix ASAP		07/29/2015
	0	0	3542593	New user form did not prevent user name less t..	Active	2-Major Bug	1-Fix ASAP		07/29/2015
	0	0	3543512	Login Page Field Validation - High Level Scenario	Active				10/31/2016

Synchronize the QA Complete with JIRA Project Management Tool

To synchronize the QA complete test management tool with the JIRA project management tool we must install and configure QA Complete – JIRA Connector.

QA Complete - JIRA Connector

It is a connector to synchronize JIRA issues with QA Complete defects or requirements.it synchronizes the data based on the settings that are done i.e., Unidirectional synchronization/bidirectional synchronization.

- **Bidirectional –** any change made in one system will be updated in the other system
- **Unidirectional –** changes that are made in the source system will be updated in the destination system (changes made in the destination system will be overwritten by the changes made later in the source system).

Advantages

1. Defects are created automatically under the defects section as a test step fails in a test set.
2. Automation tests can be scheduled accordingly for the next run.
3. Synchronization with tools like JIRA helps in providing the deliverables in the right way.

Conclusion

QA Complete test management tool might be of great use to the QA team in producing effective deliverables at the right time.

EXPERIMENT 9

Aim : (a) Learn how to raise and report Bugs using Bug Tracking Tool - **Bugzilla**

Introduction

Bugzilla is an open-source issue/bug tracking system that allows developers to keep track of outstanding problems with their products. It is written in Perl and uses the MYSQL database.

Bugzilla is a Defect tracking tool; however, it can be used as a test management tool as such it can be easily linked with other Test Case management tools like Quality Centre, Test link, etc.

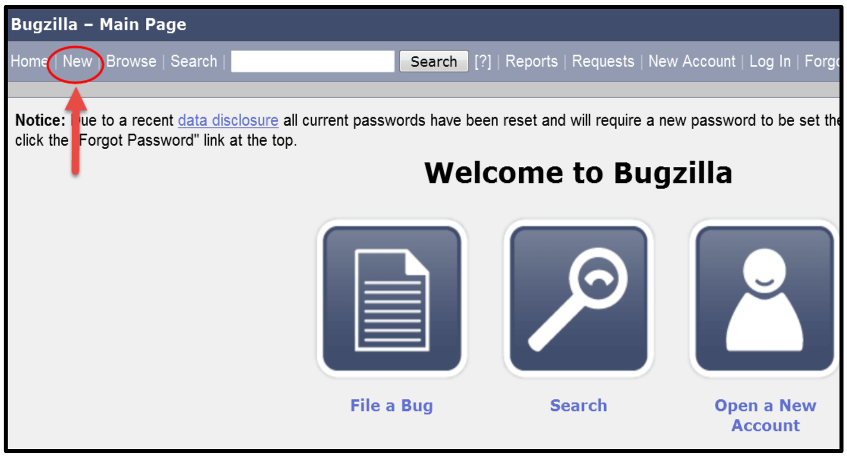
This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data - management chain.

Key Features

- Bugzilla is powerful and it has advanced searching capabilities.
- Bugzilla supports user-configurable email notifications whenever the bug status changes.
- Bugzilla displays the complete bug change history.
- Bugzilla provides an inter bug dependency track and graphic representation.
- Bugzilla allows users to attach Bug supportive files and manage them.
- Bugzilla has an integrated, product-based, granular security schema that makes it more secure.
- It has a complete security audit and runs under Perl's taint mode.
- Bugzilla supports a robust, stable RDBMS (Rational Data Base Management System) back end.
- It supports Web, XML, E-Mail, and console interfaces.
- Bugzilla has a wide range of customized, user preferences features.
- It supports a localized web user interface.
- Bugzilla has a smooth upgrade pathway among different versions.

Steps for Creating a Bug Report in Bugzilla

Step 1: To create a new bug in Bugzilla, visit the home-page of Bugzilla and click on the NEW tab from the main.


Bugzilla – Main Page

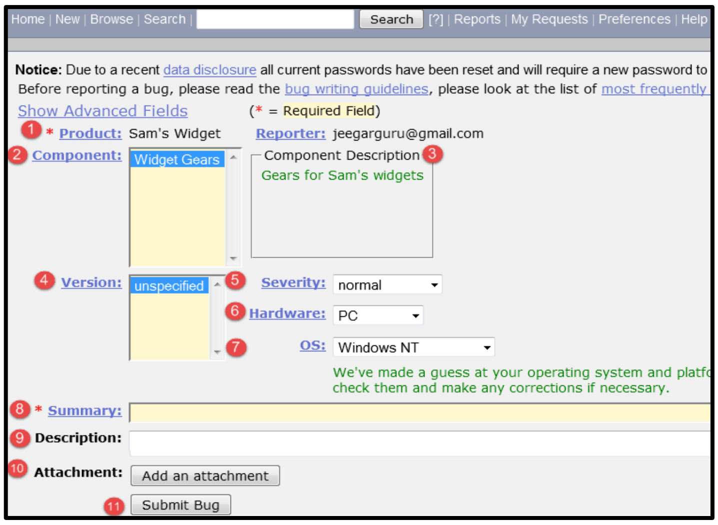
Notice: Due to a recent data disclosure all current passwords have been reset and will require a new password to be set the click the "Forgot Password" link at the top.

Welcome to Bugzilla

Home | New | Browse | Search | Search | [?] | Reports | Requests | New Account | Log In | Forg
File a Bug
Search
Open a New Account

Step 2: In the next window

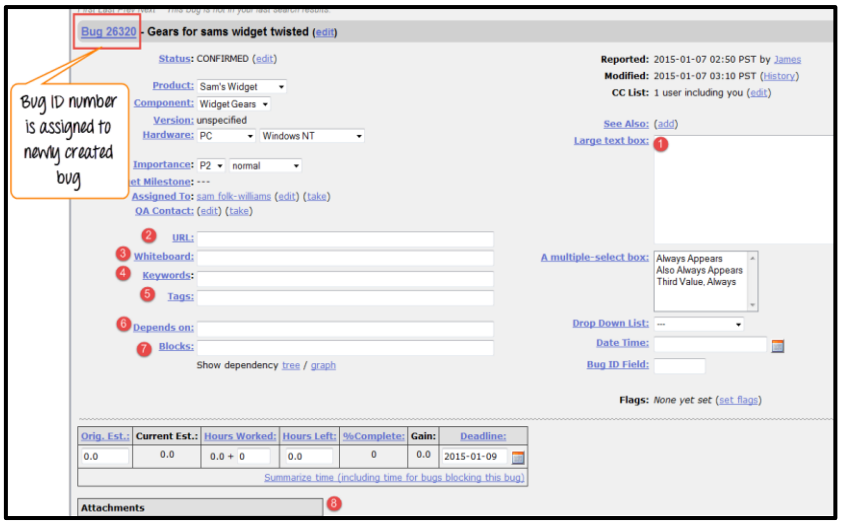
i. Enter Product -> Enter Component-> Give Component description-> Select version->Select severity ->Select Hardware-> Select OS-> Enter Summary-> Enter Description-> Attach Attachment
ii. Submit


Home | New | Browse | Search | Search | [?] | Reports | My Requests | Preferences | Help
1
2
3
4
5
6
7
8
9
10
11
Add an attachment
Submit Bug

Notice: Due to a recent data disclosure all current passwords have been reset and will require a new password to
Before reporting a bug, please read the bug writing guidelines, please look at the list of most frequently
Show Advanced Fields (* = Required Field)

* **Product:** Sam's Widget **Reporter:** jeegarguru@gmail.com
Component: Widget Gears
Component Description
Gears for Sam's widgets
Version: unspecified
Severity: normal
Hardware: PC
OS: Windows NT
We've made a guess at your operating system and platf
check them and make any corrections if necessary.
* **Summary:**
Description:
Attachment:

Step 3: The bug is created ID# 26320 is assigned to our Bug. You can also add additional information to the assigned bug-like URL, keywords, whiteboard, tags, etc. This extra information is helpful to give more detail about the Bug you have created.



Bug 26320 - Gears for sams widget twisted (edit)

Status: CONFIRMED (edit)
Product: Sam's Widget
Component: Widget Gears
Version: unspecified
Hardware: PC Windows NT
Importance: P2 normal
et Milestone: ---
Assigned To: sam folk-williams (edit) (take)
QA Contact: (edit) (take)
URL:
Whiteboard:
Keywords:
Tags:
Depends on:
Blocks:
Show dependency tree / graph

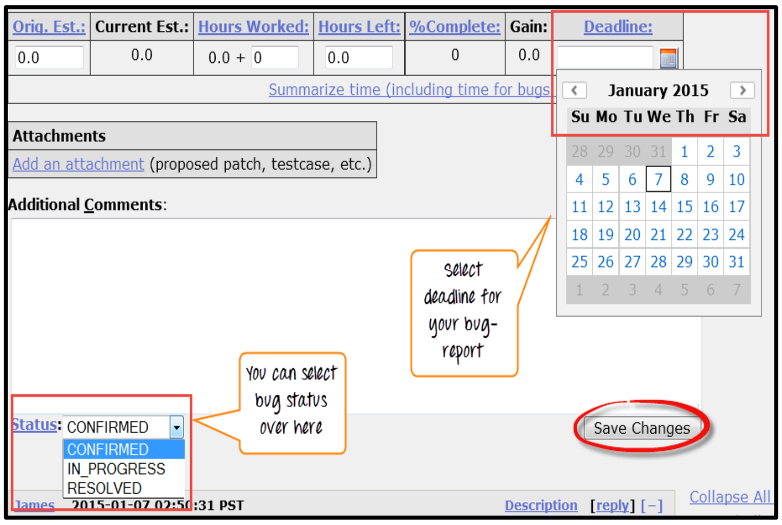
Reported: 2015-01-07 02:50 PST by James
Modified: 2015-01-07 03:10 PST (History)
CC List: 1 user including you (edit)
See Also: (add)
Large text box:
A multiple-select box: Always Appears / Also Always Appears / Third Value, Always
Drop Down List: ---
Date Time:
Bug ID Field:
Flags: *None yet set* (set flags)

Orig. Est.:	Current Est.:	Hours Worked:	Hours Left:	%Complete:	Gain:	Deadline:
0.0	0.0	0.0 + 0	0.0	0	0.0	2015-01-09

Summarize time (including time for bugs blocking this bug)

Attachments

Step 4: In the same window if you scroll down further. You can select the deadline date and also the status of the bug. The deadline in Bugzilla usually gives the time-limit to resolve the bug in a given time frame.



Orig. Est.:	Current Est.:	Hours Worked:	Hours Left:	%Complete:	Gain:	Deadline:
0.0	0.0	0.0 + 0	0.0	0	0.0	

Summarize time (including time for bugs

Attachments
Add an attachment (proposed patch, testcase, etc.)

Additional Comments:

Status: CONFIRMED
CONFIRMED
IN_PROGRESS
RESOLVED

James 2015-01-07 02:50:31 PST Description [reply] [−]

Aim : (b) To study Open Source Testing Tool **- SELENIUM**

Introduction

Selenium is one of the most widely used open- source Web UI automation testing suite. It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms, and programming languages. Selenium can be easily deployed on platforms such as Windows, Linux, Solaris, and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile, and android.

Selenium supports a variety of programming languages through the use of drivers specific to each language. Languages supported by Selenium include C#, Java, Perl, PHP, Python, and Ruby. Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome, and Safari.

Features

- Selenium is an open-source and portable Web testing Framework.
- Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
- It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
- It also supports parallel test execution which reduces time and increases the efficiency of tests.
- Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
- Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
- Selenium requires fewer resources as compared to other automation test tools.
- WebDriver API has been indulged in selenium which is one of the most important modifications done to selenium.

Selenium Tool Suite

Selenium is not just a single tool but a suite of software, each with a different approach to supporting automation testing. It comprises four major components which include:

1. Selenium Integrated Development Environment (IDE)

Selenium IDE is implemented as a Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit, and TestNG. You can use this exported script in Selenium RC or Webdriver. Selenium IDE has limited scope and the generated test scripts are not very robust and portable.

2. Selenium Remote Control

Selenium RC (officially deprecated by selenium) allows testers to write automated web application UI tests in any of the supported programming languages. It also involves an HTTP proxy server which enables the browser to believe that the web application being tested comes from the domain provided by the proxy server. Selenium RC comes with two components:

- Selenium RC Server (acts as an HTTP proxy for web requests).
- Selenium RC Client (library containing your programming language code).

3. Selenium WebDriver

Selenium WebDriver (Selenium 2) is the successor to Selenium RC and is by far the most important component of Selenium Suite. Selenium WebDriver provides a programming interface to create and execute test cases. Test scripts are written to identify web elements on web pages and then desired actions are performed on those elements.

Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the web browser. Since WebDriver directly calls the methods of different browsers, hence we have separate drivers for each browser. Some of the most widely used web drivers include:

- Mozilla Firefox Driver (Gecko Driver)
- Google Chrome Driver
- Internet Explorer Driver
- Opera Driver
- Safari Driver
- HTML Unit Driver (a special headless driver)

4. Selenium Grid

Selenium Grid is also an important component of Selenium Suite which allows us to run our tests on different machines against different browsers in parallel. In simple words, we can run our tests simultaneously on different machines running different browsers and operating systems.

Selenium Grid follows the Hub-Node Architecture to achieve parallel execution of test scripts. The Hub is considered as the master of the network and the other will be the nodes. Hub controls the execution of test scripts on various nodes of the network.

Limitations

- Selenium does not support automation testing for desktop applications.
- Selenium requires high skill sets to automate tests more effectively.
- Since Selenium is open-source software, you have to rely on community forums to get your technical issues resolved.
- We can't perform automation tests on web services like SOAP or REST using Selenium.
- We should know at least one of the supported programming languages to create test scripts in Selenium WebDriver.

17/11/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 10

Aim : (a) Write a program in C/C++ to find the largest of 3 numbers. The test suite selected by a testing technique is given below-

S. No.	A	B	C	Expected O/P
1	6	10	2	10
2	10	6	2	10
3	6	2	10	10
4	6	10	20	20

Create 5 Mutants (M1 to M5) and calculate Mutation Score of this test suite.

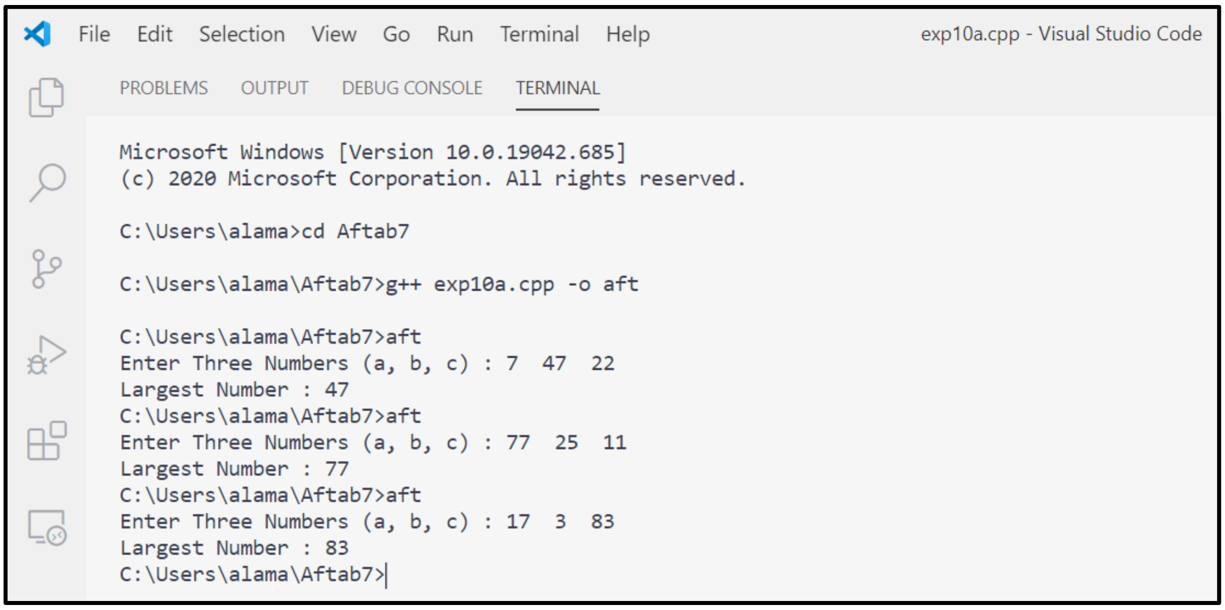
Algorithm :

- Take three numbers a, b and c as input from the user.
- If (a > = b) and (a > = c) then Largest number is a.
- Else If (b > = a) and (b > = c) then Largest number is b.
- Else the Largest number is c.

Code :

```
int main()
{
    int a, b, c;
    cout << "Enter Three Numbers (a, b, c) : ";
    cin >> a >> b >> c;
    if ((a >= b) && (a >= c))
        cout << "Largest Number : " << a;
    else if (b >= c)
        cout << "Largest Number : " << b;
    else
        cout << "Largest Number : " << c;
    return 0;
}
```

Output Screenshot :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp10a.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter Three Numbers (a, b, c) : 7  47  22
Largest Number : 47
C:\Users\alama\Aftab7>aft
Enter Three Numbers (a, b, c) : 77  25  11
Largest Number : 77
C:\Users\alama\Aftab7>aft
Enter Three Numbers (a, b, c) : 17  3  83
Largest Number : 83
C:\Users\alama\Aftab7>
```

Mutation Testing : We will create Mutants for the above program as:

- ***Mutant 1 – M1***

```
int main()
{
int a, b, c;
cout << "Enter Three Numbers (a, b, c) : ";
cin >> a >> b >> c;
if ((c >= a) && (c <= b)) //changing the if condition in terms of c
    cout << "Largest Number : " << a;
else if (b >= c)
    cout << "Largest Number : " << b;
else
    cout << "Largest Number : " << c;
return 0;
}
```

- ***Mutant 2 – M2***

```
int main()
{
int a, b, c;
cout << "Enter Three Numbers (a, b, c) : ";
cin >> a >> b >> c;
if ((a == b) && (a >= c)) //replacing >= by ==
    cout << "Largest Number : " << a;
else if (b >= c)
    cout << "Largest Number : " << b;
else
    cout << "Largest Number : " << c;
return 0;
}
```

- ***Mutant 3 – M3***

```
int main()
{
int a, b, c;
cout << "Enter Three Numbers (a, b, c) : ";
cin >> a >> b >> c;
if ((a >= b) && (a >= c)) //replacing && by ||
    cout << "Largest Number : " << a;
else if (b >= c)
    cout << "Largest Number : " << b;
else
    cout << "Largest Number : " << c;
return 0;
}
```

- ***Mutant 4 – M4***

```cpp
int main()
{
int a, b, c;
cout << "Enter Three Numbers (a, b, c) : ";
cin >> a >> b >> c;
if ((a >= b) && (a >= c))
    cout << "Largest Number : " << c; //replacing a by c
else if (b >= c)
    cout << "Largest Number : " << b;
else
    cout << "Largest Number : " << c;
return 0;
}
```

- ***Mutant 5 – M5***

```cpp
int main()
{
int a, b, c;
cout << "Enter Three Numbers (a, b, c) : ";
cin >> a >> b >> c;
if ((b >= a) && (b >= c)) //changing the if condition in terms of b
    cout << "Largest Number : " << a;
else if (b >= c)
    cout << "Largest Number : " << b;
else
    cout << "Largest Number : " << c;
return 0;
}
```

Mutants Test Cases Table

Mutant	Test Case	Input	Expected Output	Mutant Output	Test Result	Remark
M1	1	<6,10,2>	10	10	Fail	Killable
	2	<10,6,2>	10	Program Terminates	Pass	Killed
	3	<6,2,10>	10	10	Fail	Killable
	4	<6,10,20>	20	20	Fail	Killable
M2	1	<6,10,2>	10	10	Fail	Killable
	2	<10,6,2>	10	Program Terminates	Pass	Killed
	3	<6,2,10>	10	10	Fail	Killable
	4	<6,10,20>	20	20	Fail	Killable
M3	1	<6,10,2>	10	Program Terminates	Pass	Killed
	2	<10,6,2>	10	10	Fail	Killable
	3	<6,2,10>	10	Program Terminates	Pass	Killed
	4	<6,10,20>	20	20	Fail	Killable
M4	1	<6,10,2>	10	10	Fail	Killable
	2	<10,6,2>	10	Program Terminates	Pass	Killed
	3	<6,2,10>	10	10	Fail	Killable
	4	<6,10,20>	20	20	Fail	Killable
M5	1	<6,10,2>	10	10	Fail	Killable
	2	<10,6,2>	10	Program Terminates	Pass	Killed
	3	<6,2,10>	10	10	Fail	Killable
	4	<6,10,20>	20	20	Fail	Killable

Mutation Score

$$\boldsymbol{Mutation\ Score} = \frac{\mathbf{100} * \boldsymbol{No.of\ Killed\ Mutants}}{(\boldsymbol{No.of\ Total\ Mutants} - \boldsymbol{No.of\ Equivalent\ Mutants})}$$

Here,

No. of Killed Mutants	=	5
No. of Total Mutants	=	5
No. of Equivalent Mutants	=	0

So,

Mutation Score	=	100 * 5 / (5 - 0)
	=	100 * 5 / 5
	=	**100**

17/11/2020 **Mohd. Aftab Alam**
02013302717

Aim : (b) Write a program in C/C++ to determine the day of the given date and perform Slice-based testing for all variables.

Algorithm :

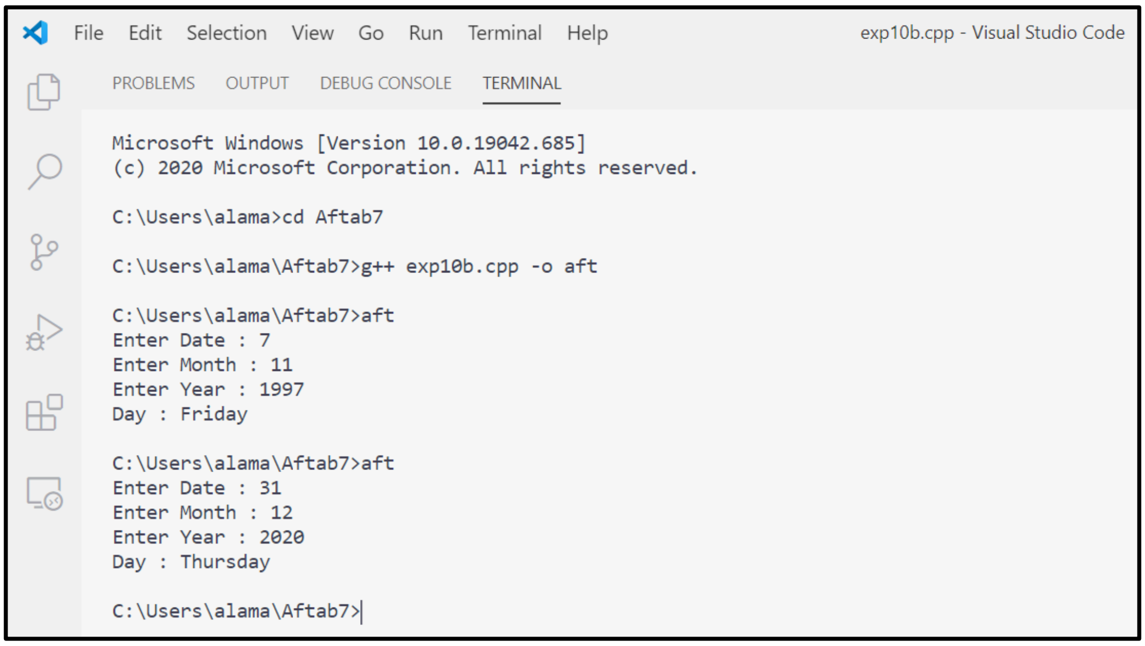
- Take three inputs from the user for day, month, and year.
- Calculate the day of the given date using the formula:

  Day = (d + m + y + [y / 4] + c) mod 7
- Here, c stands for the century number.
- Find the day according to the calculated number of the day.
- Print the day of the given date as calculated.

Code :

```
1. int main()
2. {
3. int d, m, y;
4. cout << "Enter Date : ";
5. cin >> d;
6. cout << "Enter Month : ";
7. cin >> m;
8. cout << "Enter Year : ";
9. cin >> y;
10.const char *Names[] = {"Sunday", "Monday", "Tuesday", "Wednesday",
   "Thursday", "Friday", "Saturday"};
11.int day = 0;
12.static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
13.y -= m < 3;
14.day = (y + y / 4 - y / 100 + y / 400 + t[m - 1] + d) % 7;
15.cout << "Day : " << Names[day] << endl;
16.return 0;
17.}
```

Output Screenshot :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp10b.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter Date : 7
Enter Month : 11
Enter Year : 1997
Day : Friday

C:\Users\alama\Aftab7>aft
Enter Date : 31
Enter Month : 12
Enter Year : 2020
Day : Thursday

C:\Users\alama\Aftab7>
```

Slice Based Testing :

There is total 6 variables in the program. We can create slices for each of them.

- ***Variable: d***

S(d,5) / S(d,17) = {1-5,17}

```
1. int main()
2. {
3. int d, m, y;
4. cout << "Enter Date : ";
5. cin >> d;
17.}
```

- ***Variable: m***

S(m,7) / S(m,17) = {1-3,6,7,17}

```
1. int main()
2. {
3. int d, m, y;
6. cout << "Enter Month : ";
7. cin >> m;
17.}
```

- ***Variable: y***

S(y,9) / S(y,17) = {1-3,8,9,17}

```
1. int main()
2. {
3. int d, m, y;
8. cout << "Enter Year : ";
9. cin >> y;
17.}
```

S(y,13) / S(y,17) = {1-3,8,9,13,17}

```
1. int main()
2. {
3. int d, m, y;
8. cout << "Enter Year : ";
9. cin >> y;
13.y -= m < 3;
17.}
```

- ***Variable: Names***

S(Names,10) = {1-2,10,17}

```
1. int main()
2. {
10.const char *Names[] = {"Sunday", "Monday", "Tuesday", "Wednesday",
   "Thursday", "Friday", "Saturday"};
17.}
```

S(Names,15) / S(Names,17) = {1-17}

```
1. int main()
2. {
3. int d, m, y;
4. cout << "Enter Date : ";
5. cin >> d;
6. cout << "Enter Month : ";
7. cin >> m;
8. cout << "Enter Year : ";
9. cin >> y;
10.const char *Names[] = {"Sunday", "Monday", "Tuesday", "Wednesday",
   "Thursday", "Friday", "Saturday"};
11.int day = 0;
12.static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
13.y -= m < 3;
14.day = (y + y / 4 - y / 100 + y / 400 + t[m - 1] + d) % 7;
15.cout << "Day : " << Names[day] << endl;
16.return 0;
17.}
```

- ***Variable: day***

S(day,11) = {1-2,11,17}

```
1. int main()
2. {
11.int day = 0;
17.}
```

S(day,14) / S(day,17) = {1-14,16,17}

```
1. int main()
2. {
3. int d, m, y;
4. cout << "Enter Date : ";
5. cin >> d;
6. cout << "Enter Month : ";
7. cin >> m;
8. cout << "Enter Year : ";
9. cin >> y;
10.const char *Names[] = {"Sunday", "Monday", "Tuesday", "Wednesday",
   "Thursday", "Friday", "Saturday"};
11.int day = 0;
12.static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
13.y -= m < 3;
14.day = (y + y / 4 - y / 100 + y / 400 + t[m - 1] + d) % 7;
16.return 0;
17.}
```

- ***Variable: t***

S(t,12) / S(t,17) = {1-2,12,17}

```
1. int main()
2. {
12.static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
17.}
```

Test Cases :

S. No.	**Slice**	**Lines Covered**	**Variables**			**Expected Output**
			d	**m**	**y**	
1	S(d,5) / S(d,17)	1-5,17	12			No Output
2	S(m,7) / S(m,17)	1-3,6,7,17		4		No Output
3	S(y,9)	1-3,8,9,17			2019	No Output
4	S(y,13) / S(y,17)	1-3,8-9,1317			2019	No Output
5	S(Names,10)	1-2,10,17				No Output
6	S(Names,15) / S(Names,17)	1-17	12	4	2019	Friday
7	S(day,11)	1-2,11,17				No Output
8	S(day,14) / S(day,17)	1-14,16,17	12	4	2019	No Output
9	S(t,12) / S(t,17)	1,2,12,17				No Output

21/11/2020 **Mohd. Aftab Alam**
02013302717

EXPERIMENT 11

Aim : (a) Write a Program in C/C++ to compute total salary of an employee when his basic salary is given. (Given: HRA = 3% of basic, DA = 8% of basic, CCA/MA = Rs. 100, Tax = Rs. 300, PF = Rs.780, TA = Rs. 800). Perform Slice based testing for all variables.

$$\textit{Total Salary} = (\textit{Basic} + \textit{HRA} + \textit{DA} + \textit{TA}) - (\textit{Tax} + \textit{CM} + \textit{PF})$$

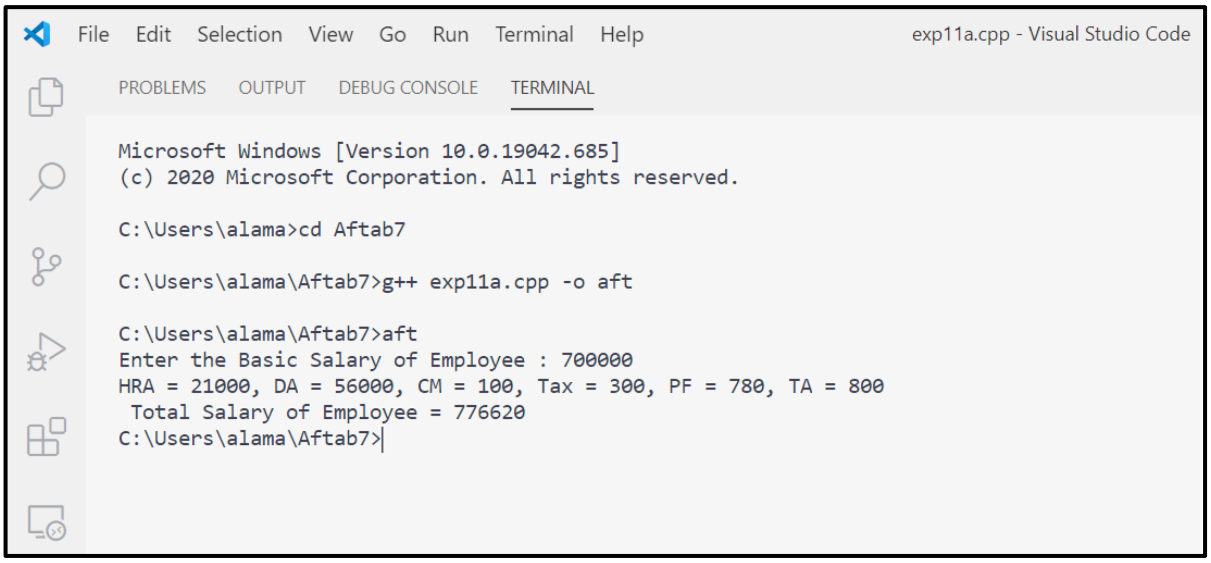
Algorithm :

- Take the Basic Salary of the employee as input from the user.
- Calculate HRA and DA using the basic salary.
- Calculate the Total Salary by combining all the values.
- Print the Total Salary of the employee as calculated on the screen.

Code :

```
#include <iostream>
using namespace std;
int main() {
float basic, HRA, DA, CM, tax, PF, TA, total_salary;
printf("Enter the Basic Salary of Employee : ");
cin >> basic;
HRA = (basic * 3) / 100;
DA = (basic * 8) / 100;
CM = 100;
tax = 300;
PF = 780;
TA = 800;
cout<<"HRA="<<HRA<<"DA="<<DA<<"CM="<<CM<<"Tax="<<tax<<"PF="<<PF<<"TA="<<TA;
total_salary = (basic + HRA + DA + TA) - (tax + CM + PF);
cout << "\n Total Salary of Employee = " << total_salary;
return 0;
}
```

Output Screenshot :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp11a.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter the Basic Salary of Employee : 700000
HRA = 21000, DA = 56000, CM = 100, Tax = 300, PF = 780, TA = 800
 Total Salary of Employee = 776620
C:\Users\alama\Aftab7>
```

21/11/2020 **Mohd. Aftab Alam**
02013302717

Slice Based Testing :

There is total 8 variables in the program. We can create slices for each of them.

- ***Variable: basic***

S(basic,5) / S(basic,15) = {1-5,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
15.}
```

- ***Variable: HRA***

S(HRA,6) / S(HRA,15) = {1-6,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
15.}
```

- ***Variable: DA***

S(DA,6) / S(DA,15) = {1-6,7,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
6. DA = (basic * 8) / 100;
15.}
```

- ***Variable: CM***

S(CM,8) / S(CM,15) = {1-3,8,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
7. CM = 100;
15.}
```

- ***Variable: tax***

S(tax,8) / S(tax,15) = {1-3,8,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
8. tax = 300;
15.}
```

- ***Variable: PF***

S(PF,10) / S(PF,15) = {1-3,10,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
9. PF = 780;
15.}
```

- ***Variable: TA***

S(TA,11) / S(TA,15) = {1-3,11,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
10.TA = 800;
15.}
```

- ***Variable: total_salary***

S(Total,12) = {1-12,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf("Enter the Basic Salary of Employee : ");
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10.TA = 800;
12.total_salary = (basic + HRA + DA + TA) - (tax + CM + PF);
15.}
```

S(Total,13) / S(Total,15) = {1-13,14,15}

```
1. int main() {
2. float basic, HRA, DA, CM, tax, PF, TA, total_salary;
3. printf(“Enter the Basic Salary of Employee : “);
4. cin >> basic;
5. HRA = (basic * 3) / 100;
6. DA = (basic * 8) / 100;
7. CM = 100;
8. tax = 300;
9. PF = 780;
10. TA = 800;
11.cout<<”HRA=”<<HRA<<”DA=”<<DA<<”CM=”<<CM<<”Tax=”<<tax<<”PF=”<<PF<<”TA=”<<TA;
12. total_salary = (basic + HRA + DA + TA) – (tax + CM + PF);
13. cout << “\n Total Salary of Employee = “ << total_salary;
14. return 0;
15. }
```

Test Cases :

S. No.	Slice	Lines Covered	Variables			Expected Output
			Basic	HRA	DA	
1	S(Basic,5) / S(Basic,15)	1-5,15	1000	30	80	No Output
2	S(HRA,6) / S(HRA,15)	1-6,15	3000	90	240	No Output
3	S(DA,7) / S(DA,15)	1-5,7,15	3000	90	240	No Output
4	S(MA,8) / S(MA,15)	1-3,8,15	3000	90	240	No Output
5	S(ITAX,9) / S(ITAX,15)	1-3,9,15	3000	90	240	No Output
6	S(PF,10) / S(PF,15)	1-3,10,15	3000	90	240	No Output
7	S(TA,11) / S(TA,15)	1-3,11,15	3000	90	240	No Output
8	S(Total,12)	1-12,15	3000	90	240	No Output
9	S(Total,13) / S(Total,15)	1-13,14,15	5000	150	400	7530

21/11/2020 **Mohd. Aftab Alam**
02013302717

Aim :

(b) Write a Program in C/C++ to find whether a triangle is right, acute or obtuse angled. Its input is a triplet of 3 positive integers (say a, b, c) from interval (1 to 100). Perform slice-based testing for all variables.

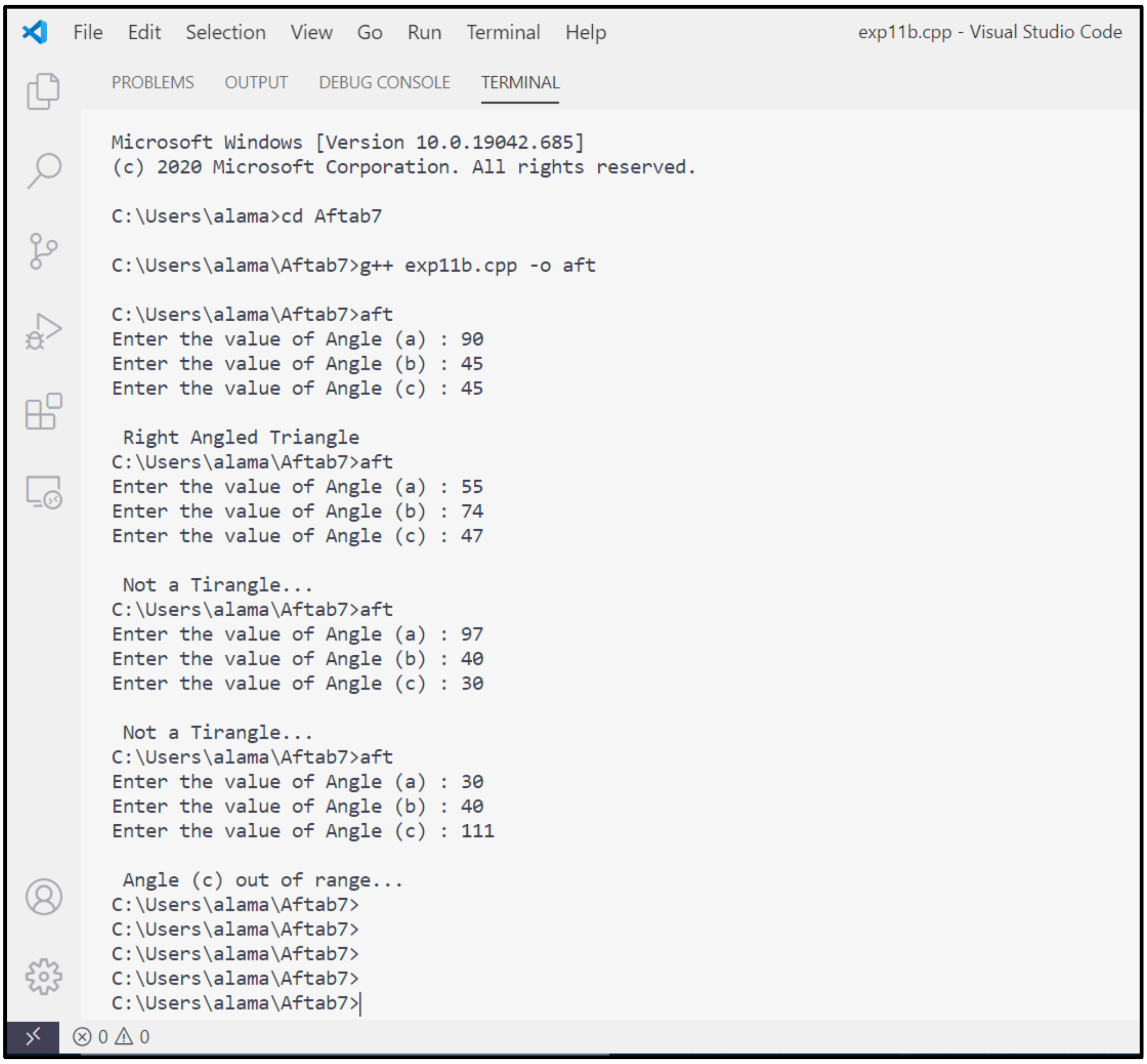
Algorithm :

- Take three input from the user for the angles a, b and c.
- Check whether they lie in the given interval.
- Check the sum of all the angles and if that is equal to 180° then proceed further, else it is not a triangle.
- After checking the validity of triangle check for following conditions and classify them into categories:
    - If any angle is 90°, then it is a Right-Angled Triangle.
    - If any angle is greater than 90°, then it is Obtuse Angled Triangle.
    - If any angle is smaller than 90°, then it is Acute Angled Triangle.

Code:

```cpp
#include <iostream>
using namespace std;
int main() {
    int a, b, c, sum;
    cout << "Enter the value of Angle (a) : ";
    cin >> a;
    cout << "Enter the value of Angle (b) : ";
    cin >> b;
    cout << "Enter the value of Angle (c) : ";
    cin >> c;
    if (a > -1 && a < 101) {
        if (b > -1 && b < 101) {
            if (c > -1 && c < 101) {
                sum = a + b + c;
                if (sum == 180) {
                    if (a == 90 || b == 90 || c == 90) {
                        cout << "\n Right Angled Triangle";
                    }
                    else if (a > 90 || b > 90 || c > 90) {
                        cout << "\n Obtuse Angled Triangle";
                    }
                    else if (a < 90 && b < 90 && c < 90) {
                        cout << "\n Acute Angled Triangle";
                    }
                }
                else {
                    cout << "\n Not a Tirangle...";
                }
            }
            else {
                cout << "\n Angle (c) out of range...";
            }
        }
        else {
            cout << "\n Angle (b) out of range...";
        }
    }
    else
    {
        cout << "\n Angle (a) out of range...";
    }
    return 0;
}
```

Output Screenshot :



```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alama>cd Aftab7

C:\Users\alama\Aftab7>g++ exp11b.cpp -o aft

C:\Users\alama\Aftab7>aft
Enter the value of Angle (a) : 90
Enter the value of Angle (b) : 45
Enter the value of Angle (c) : 45

 Right Angled Triangle
C:\Users\alama\Aftab7>aft
Enter the value of Angle (a) : 55
Enter the value of Angle (b) : 74
Enter the value of Angle (c) : 47

 Not a Tirangle...
C:\Users\alama\Aftab7>aft
Enter the value of Angle (a) : 97
Enter the value of Angle (b) : 40
Enter the value of Angle (c) : 30

 Not a Tirangle...
C:\Users\alama\Aftab7>aft
Enter the value of Angle (a) : 30
Enter the value of Angle (b) : 40
Enter the value of Angle (c) : 111

 Angle (c) out of range...
C:\Users\alama\Aftab7>
C:\Users\alama\Aftab7>
C:\Users\alama\Aftab7>
C:\Users\alama\Aftab7>
C:\Users\alama\Aftab7>
```

21/11/2020

Mohd. Aftab Alam
02013302717

Slice Based Testing :

There is total 4 variables in the program. We can create slices for each of them.

- ***Variable: a***

S (a,4) = {1,2,3,4,43}

- ***Variable: b***

S (b,4) = {1,2,3,4,43}

- ***Variable: c***

S (c,4) = {1,2,3,4,43}

- ***Variable: sum***

S (sum,4) = {1,2,3,4,43}

S (sum,4) = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,29,34,38,41,43}