

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).

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

1. Perform data flow testing

2. Slice based testing for all variables

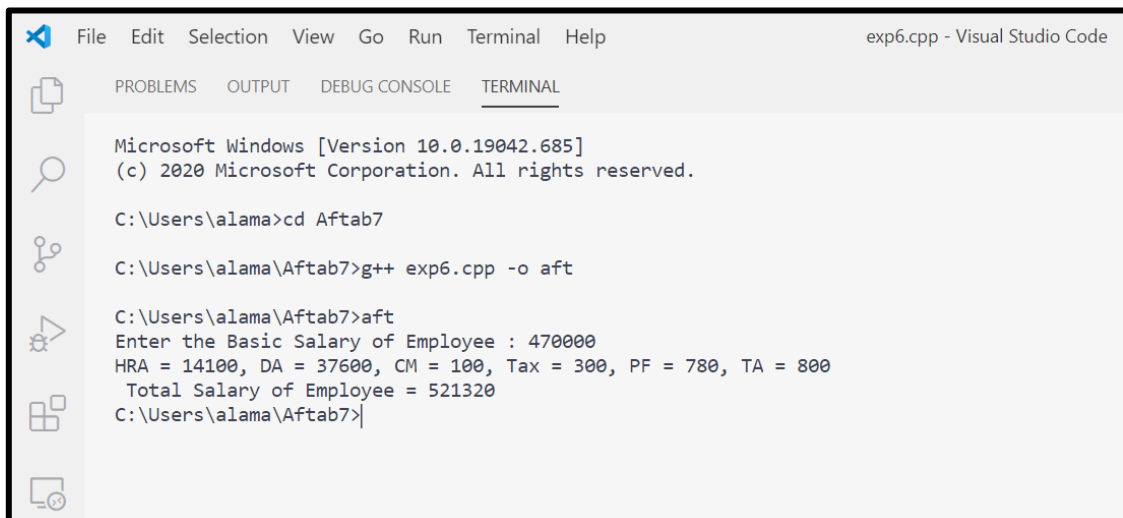
Algorithm :

- 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;
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 :



```
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>|
```

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**

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

```
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**

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

```
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**

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

```
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**

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

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

- *Variable: tax*

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

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

- *Variable: PF*

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

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

- *Variable: TA*

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

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

- *Variable: total_salary*

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

```
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.}
```

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

```

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	