**LAB EXERCISE 5**

Write a program for the following Payroll Problem

Assumptions (for all cases)

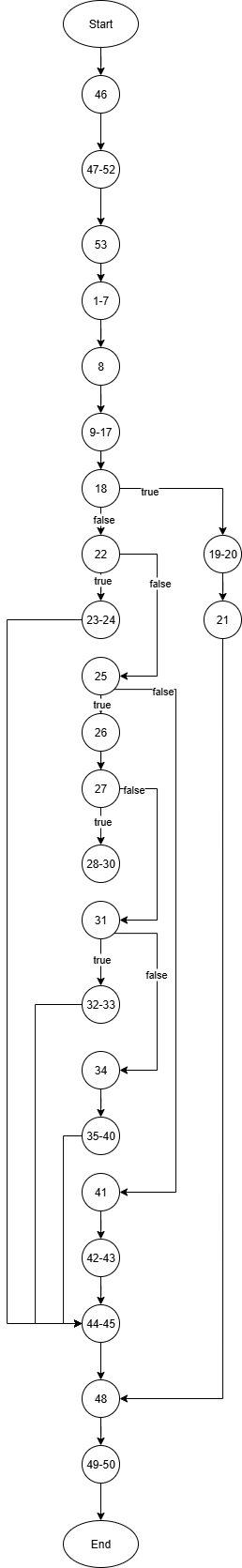
* employee\_type ∈ {S (Salaried), H (Hourly)}
* hours\_worked ≥ 0 (hours can be fractional unless specified otherwise)
* base\_salary = ₹50,000 (for salaried employees)
* hourly\_rate = ₹200 (for hourly employees)
* Overtime threshold = 40 hours
* Overtime pay = overtime\_hours × hourly\_rate × 1.5
* Outputs: total\_pay, pay\_base\_salary (Y/N), pay\_hourly (Y/N), overtime (Y/N), absence\_report (Y/N)

**Exercise 5.1: Write the program to execute the Payroll Problem**

#include <stdio.h>

1. struct **PayrollResult** {
2. double total\_pay;
3. char pay\_base;
4. char pay\_hourly;
5. char overtime;
6. char absence\_report;
7. };
8. struct **PayrollResult** **calculate\_payroll**(char employee\_type, double hours) {
9. const double base\_salary = 50000.0;
10. const double hourly\_rate = 200.0;
11. const int threshold = 40;
12. struct **PayrollResult** result;
13. result.total\_pay = 0.0;
14. result.pay\_base = 'N';
15. result.pay\_hourly = 'N';
16. result.overtime = 'N';
17. result.absence\_report = 'N';
18. if (hours < 0) {
19. **printf**("Invalid hours\n");
20. return result;
21. }
22. if (employee\_type == 'S' || employee\_type == 's') {
23. result.pay\_base = 'Y';
24. result.total\_pay = base\_salary;
25. } else if (employee\_type == 'H' || employee\_type == 'h') {
26. result.pay\_hourly = 'Y';
27. if (hours < threshold) {
28. result.total\_pay = hours \* hourly\_rate;
29. result.absence\_report = 'Y';
30. }
31. else if (hours == threshold) {
32. result.total\_pay = hours \* hourly\_rate;
33. }
34. else {
35. double regular\_pay = threshold \* hourly\_rate;
36. double overtime\_hours = hours - threshold;
37. double overtime\_pay = overtime\_hours \* hourly\_rate \* 1.5;
38. result.total\_pay = regular\_pay + overtime\_pay;
39. result.overtime = 'Y';
40. }
41. } else {
42. **printf**("Invalid employee type. Only 'S' or 'H'.\n");
43. }
44. return result;
45. }
46. int **main**() {
47. char employee\_type;
48. double hours;
49. **printf**("Enter employee type (S = Salaried, H = Hourly): ");
50. **scanf**(" %c", &employee\_type);
51. **printf**("Enter number of hours worked: ");
52. **scanf**("%lf", &hours);
53. struct **PayrollResult** result = **calculate\_payroll**(employee\_type, hours);
54. **printf**("\nPayroll Summary:\n");
55. **printf**("Total Pay: Rs. %.2f\n", result.total\_pay);
56. **printf**("Pay Base Salary: %c\n", result.pay\_base);
57. **printf**("Pay Hourly: %c\n", result.pay\_hourly);
58. **printf**("Overtime: %c\n", result.overtime);
59. **printf**("Absence Report: %c\n", result.absence\_report);
60. return 0;
61. }

**Exercise 3.2: Prepare the CFG for the program “Payroll”**

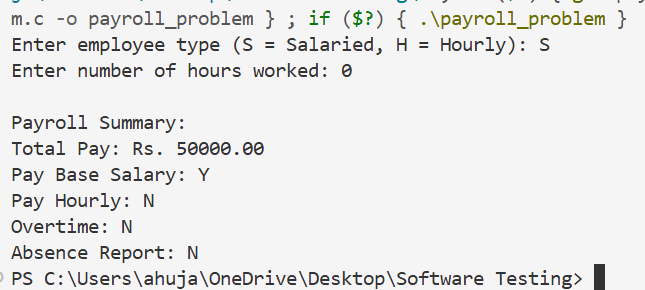


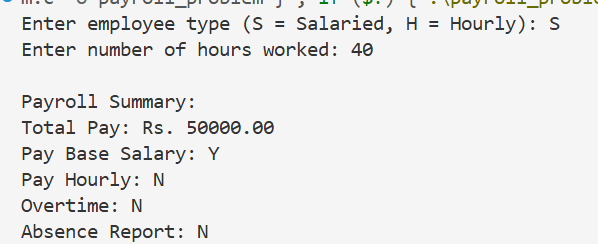
**Exercise 5.3: Write the test cases using Boundary Value, Equivalence Class and Decision Table based Testing**

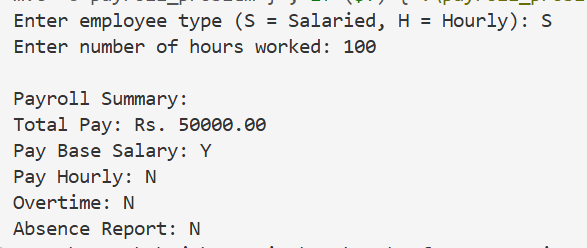
**Exercise 5.4: Execute the test cases on the Program**

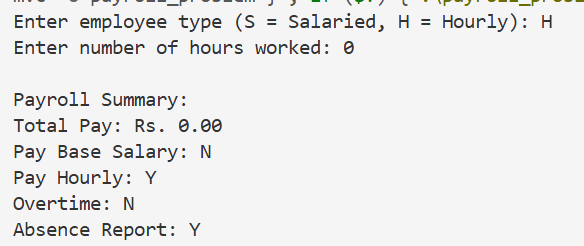
Boundary Value Analysis

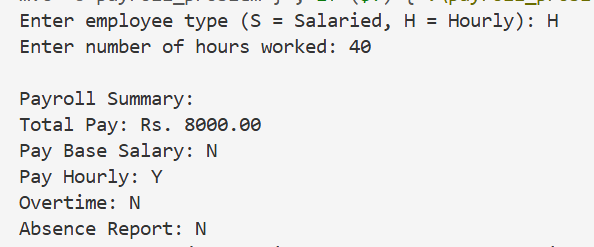
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Employee Type | Hours Worked | Expected Output | Actual Output | Result |
| 1. | Salaried | 0 | Base Salary (50000) | 50000, Base Salary | PASS |
| 2 | Salaried | 40 | Base Salary (50000) | 50000, Base Salary | PASS |
| 3 | Salaried | 100 | Base Salary (50000) | 50000, Base Salary | PASS |
| 4 | Hourly | 0 | Absence Report – Y | 0, Absence report | PASS |
| 5 | Hourly | 40 | 40\*200 | 8000.0 | PASS |
| 6 | Hourly | 41 | 40\*200 + 1\*200\*1.5 | 8300.0 | PASS |
| 7 | Hourly | 100 | 40\*200+60\*200\*1.5 | 26000.0, overtime | PASS |

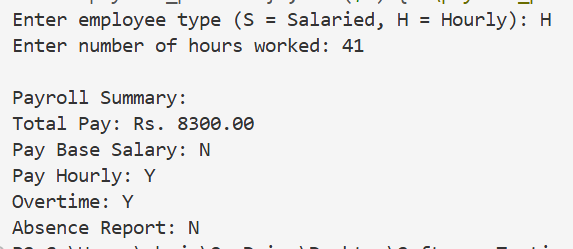


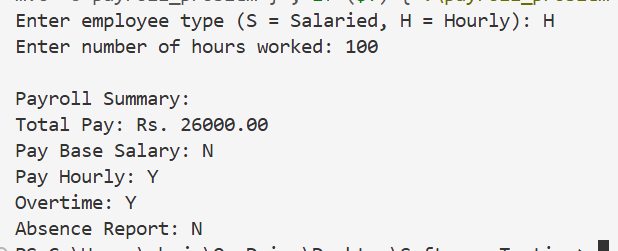










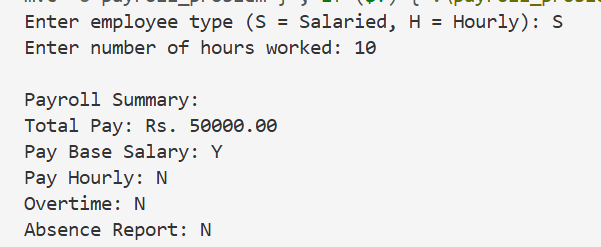


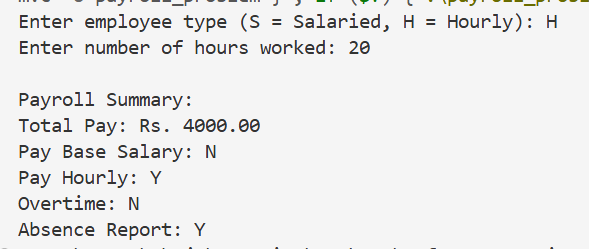
Equivalence Class Partitioning (ECP)

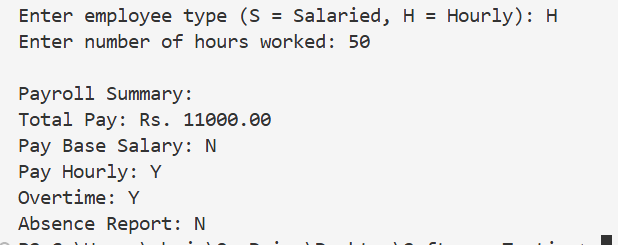
Valid/invalid partitions  
employee type: {salaried, hourly} vs invalid

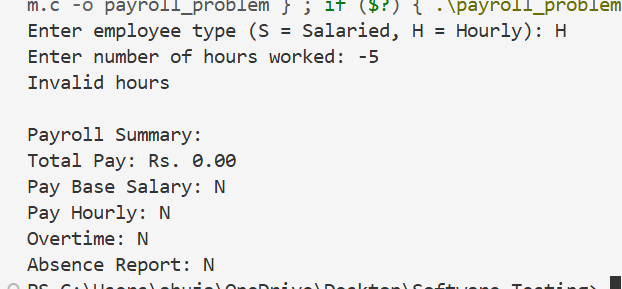
Hours: {<0(invalid), 0-40 (valid), >40 (valid overtime)}

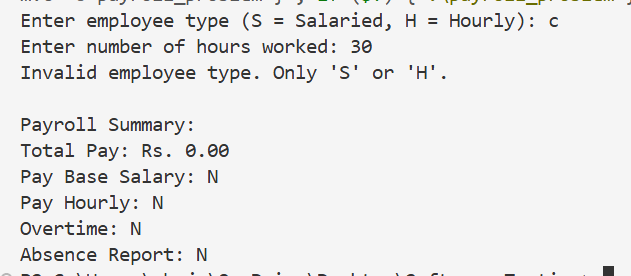
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Employee type | Hours worked | Expected output | Actual Output | Result |
| 1 | salaried | 10 | Valid – base salary | 50000 | PASS |
| 2 | hourly | 20 | Valid – 20\*200 | 4000.0 | PASS |
| 3 | hourly | 50 | Valid – 40\*200+10\*200\*1.5 | 11000.0 | PASS |
| 4 | hourly | -5 | Invalid | Invalid hours | PASS |
| 5 | c | 30 | Invalid | Invalid employee type | PASS |





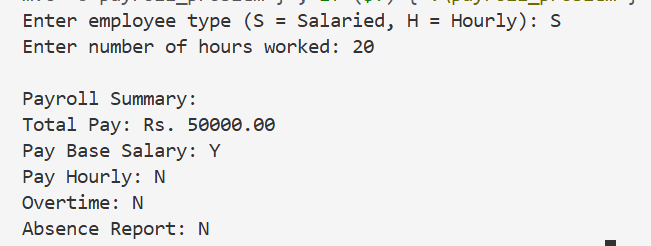


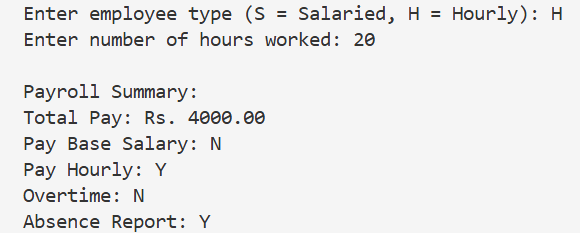


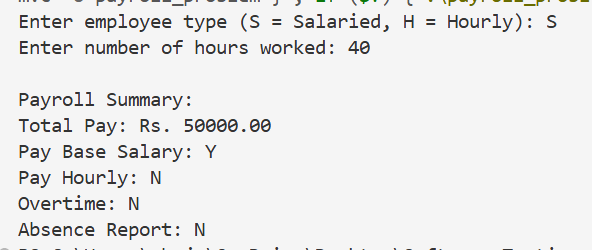


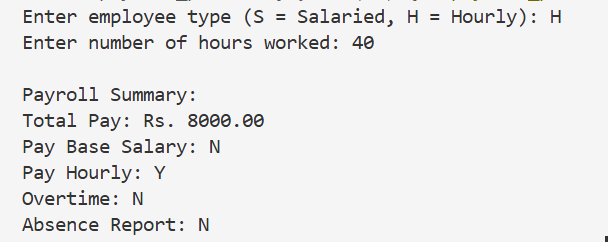
Decision Table Testing

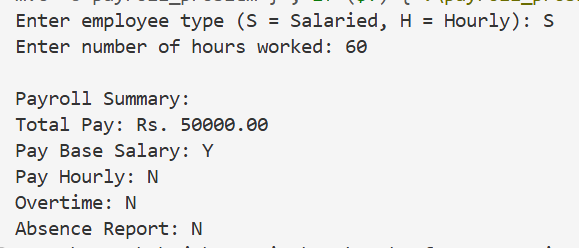
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule | Employee type | Hours Worked | Expected output | Actual output | Result |
| R1 | Salaried | 20 | 50000 | 50000.0 | PASS |
| R2 | Hourly | 20 | 20\*200, absence report | 4000.0 | PASS |
| R3 | Salaried | 40 | 50000 | 50000.0 | PASS |
| R4 | Hourly | 40 | 40\*200 | 8000.0 | PASS |
| R5 | Salaried | 60 | 50000 | 50000.0 | PASS |
| R6 | Hourly | 60 | 40\*200+20\*200\*1.5, Overtime | 14000.0 | PASS |

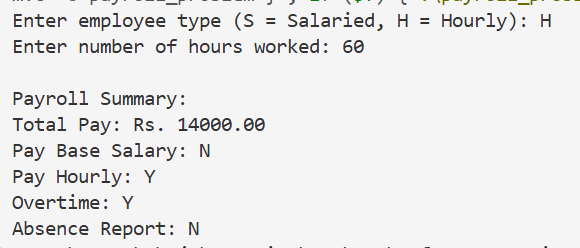












**Exercise 5.5: Debug the Program for the identified error**

The code is correct, no debugging is needed.

**Exercise 5.6: Prepare the additional test cases for the program for re-testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Employee Type | Hours Worked | Expected output | Actual output | Result |
| 1 | S | 40.5 | 50000.0 | 50000.0 | PASS |
| 2 | H | 40.5 | 40\*200+0.5\*200\*1.5 | 8150.0 | PASS |
| 3 | S | -2 | Invalid | Invalid | PASS |