Introduction to Artificial Intelligence
Lab7: Logic and Inference

Authors: Bora ILCI & Kaan Emre KARA

1. Introduction

This report presents the implementation of a Working Days Calculator in Prolog. The program calculates the date after N working days from a given date in 2024, taking into account weekends and handling month/year transitions.

2. Solution Logic Flow

Input Parsing: `parse_date/3` reads the "DDMM" string into numeric Day and Month.

Weekday Computation: `day_of_week/3` computes 0–6 offset from Jan 1, 2023 using cumulative days and mod 7.

Counting Working Days: `add_work_days/6` recursively advances one calendar day at a time via `next_day/4`, decrements N only if the new day is Mon–Fri.

Output Formatting: `format_date/3` pads day/month to two digits, and `day_name/2` maps weekday code to its name for final printout.

Code Snippet

```
% Add N working days
add_work_days(Day, Month, 0, Day, Month, WD) :-
    day_of_week(Day, Month, WD), WD {>=} 1, WD =< 5.
add_work_days(Day, Month, N, RD, RM, RWD) :-
    N > 0,
    next_day(Day, Month, ND, NM),
    day_of_week(ND, NM, WD2),
    ( WD2 >= 1, WD2 =< 5 -> M is N - 1 ; M = N ),
    add_work_days(ND, NM, M, RD, RM, RWD).
```

3. Main Components

`parse_date/3` : Splits "DDMM" into Day and Month.

`day_of_week/3` : Cumulative days before month + day minus one, mod 7.

`next_day/4` : Advances calendar by one day, handling month/year rollovers.

`add_work_days/6` : Core recursive predicate to skip weekends.

`format_date/3` & `day_name/2` : Format output string.

`n_work_days/2` : Main entry; validates N and start weekday, invokes addition and prints.

4. Challenges and Solutions

Off-by-one Errors: Ensuring correct mod calculation by subtracting one from cumulative days.

Weekend Skipping: Recursive logic correctly counts only Mon–Fri days.

Month/Year Transitions: `next_day/4` handles end of month and wrap to next year seamlessly.

Prolog I/O: Printing via `format/2` to match expected "Day, DDMM" output.

5. Limitations and Assumptions

Year Limitation
- Only works for dates in 2024
- Assumes 2024 is a leap year

Input Format
- Requires strict "DDMM" format
- No support for different date formats

Working Days
- Only considers weekends as non-working days
- No support for holidays or custom non-working days

6. Testing

The implementation includes a comprehensive test suite that covers:

- Date validation tests
- Day calculation tests
- Working days calculation tests
- Edge case tests

Example test cases:

```
?- n_work_days("2205", 6, Result).
Result = "Thursday, 3005"

?- n_work_days("0106", 10, Result).
Result = "Monday, 1706"
```

7. Conclusion

The Working Days Calculator successfully implements all required functionality using Prolog's logical programming paradigm. The solution demonstrates effective use of:

- Date manipulation and validation
- Working days calculation
- Weekend handling
- Month and year transitions

The implementation is well-documented, thoroughly tested, and handles various edge cases appropriately.