DFS Lab Assignments

Lab 1

ISI, Kolkata

August 1, 2025

Outline

Outline

Checksums

The International Standard Book Number (ISBN) is a 10-digit code that uniquely specifies a book.

The **rightmost digit** is a *checksum digit* d_1, which can be uniquely determined from the other 9 digits using the condition:

$$d_1 + 2d_2 + 3d_3 + \cdots + 10d_{10} \equiv 0 \pmod{11}$$

Notes:

- Each d_i is the i^{th} digit from the **right**.
- The checksum digit d_1 can range from 0 to 10.
- If $d_1 = 10$, it is represented as `X`.

Checksums

Example: Given the 9-digit number 020131452, the correct checksum digit is **5**, since:

$$\mathbf{5} + 2 \cdot 2 + 3 \cdot 5 + 4 \cdot 4 + 5 \cdot 1 + 6 \cdot 3 + 7 \cdot 1 + 8 \cdot 0 + 9 \cdot 2 + 10 \cdot 0 = 88$$

$$88 \equiv 0 \pmod{11}$$

Task: Write a C program that:

- Reads a 9-digit integer,
- Computes the checksum digit,
- Prints the full 10-digit ISBN in the format: 0-201-31452-5

Outline

Decision Trees: Play Decision Example

Decision trees are a widely used method for modeling decision-making processes.

The diagram below shows when you can play depending on weather artefacts. A pseudocode alongside reads these artefacts and computes the decision.

Pseudocode:

- Ask the user if it is **Cloudy**
- If user says **Yes**:
  - Print "`You can Play!`"
- Else:
  - Ask if it is **Windy**
  - If Yes: print "`Don't Play!`"
  - Else: print "`You can Play!`"

Decision Tree: Diabetic or Non-Diabetic

This task involves implementing a decision tree that determines whether a person is **Diabetic (D)** or **Non-Diabetic (ND)** based on certain health parameters.

Required inputs:

- **Glucose Level**
- **Age**

These two are always asked. Based on the values, the decision tree may request additional inputs:

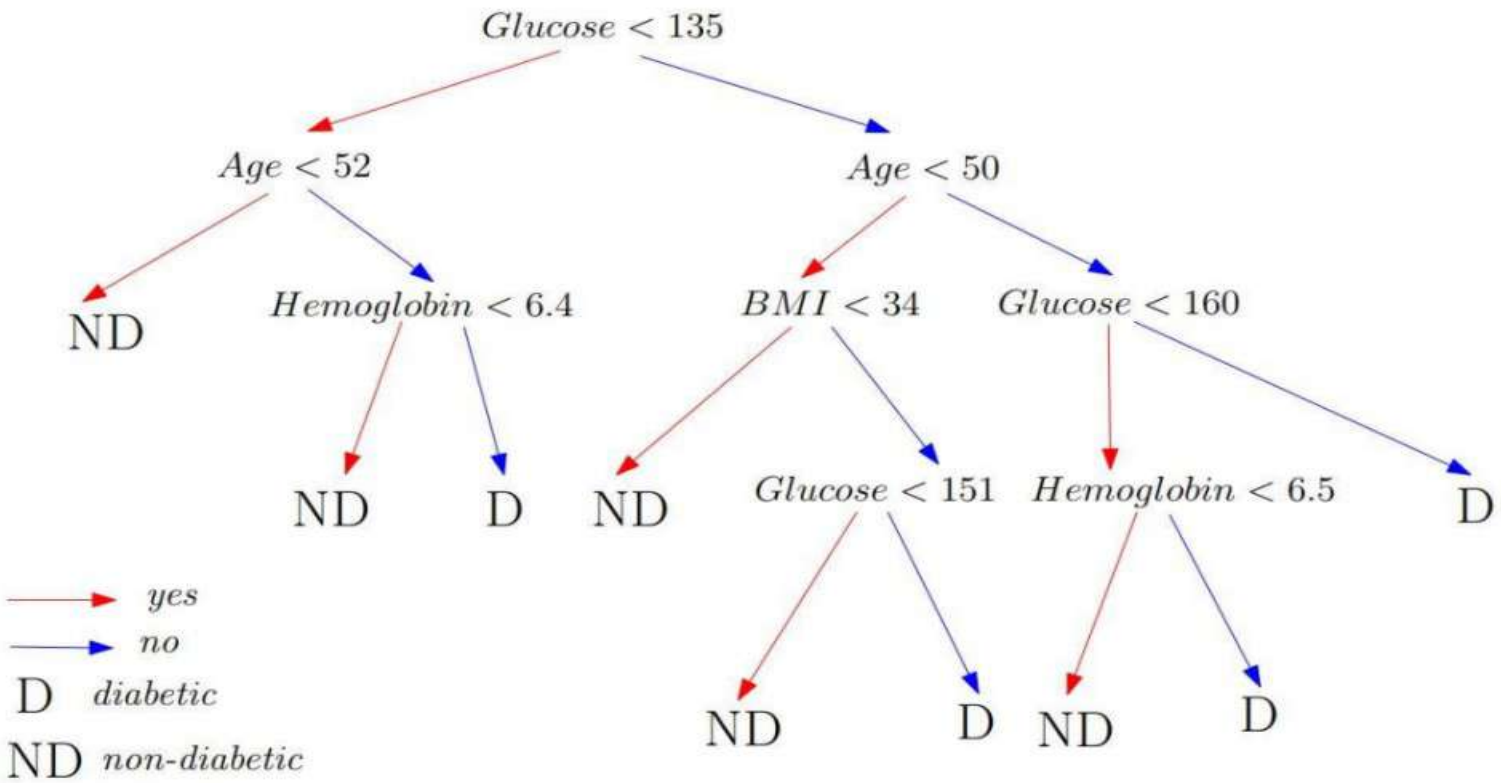
- **Body Mass Index (BMI)**
- **Haemoglobin Level**

These are conditionally prompted, depending on earlier input.

Decision Tree: Diabetic or Non-Diabetic

Goal: Write a program that:

- Asks the user for required inputs
- Navigates the decision tree logic correctly
- Outputs a clear diagnosis: **Diabetic (D)** or **Non-Diabetic (ND)**



Decision Tree: Diabetic or Non-Diabetic

Flow Overview:

- Program starts by asking for:
  - **Glucose level** and **Age**
- Based on responses, it either:
  - Reaches a diagnosis, or
  - Asks for **BMI** or **Hemoglobin**, only if needed
- All additional questions are nested within the correct `if-else` branches
- Final output: **Diabetic (D)** or **Non-Diabetic (ND)**

Outline

Number-to-English Conversion

Goal: Convert an integer (0 to 999999) into its English word form.

Word list to use:

`zero, one, two, ..., nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, hundred, thousand, lakh`

Rules:

- Always use `thousand` and `lakh` instead of expressing them as hundreds.
- Avoid phrases like `fifteen hundred`; instead use `one thousand five hundred`.
- For 0, output: `zero`.

Number-to-English Conversion

Example:

`213425` $\rightarrow$ `two lakh thirteen thousand four hundred twenty five`

Task: Write a `C program` that:

- Reads an integer between 0 and 999999.
- Prints its English equivalent following the above rules.

Outline

Inverse Permutation

Definition: An inverse permutation is one in which each number and the index it occupies are exchanged.

Example:

- Original: `a[] = { 2, 7, 4, 9, 8, 3, 5, 0, 6, 1 }`
- Inverse: `b[] = { 7, 9, 0, 5, 2, 6, 8, 1, 4, 3 }`
- Property: `a[b[i]] = i` and `b[a[i]] = i`

Task: Write a `C program` that:

- Reads a permutation of integers `0 to n-1` from the user
- Validates that it is a correct permutation
- Computes and prints its inverse

Note: A valid permutation contains all integers from `0` to `n-1` exactly once.

Outline

Decimal Expansion of Rational Numbers

Goal: Given two integers `p` and `q`, compute the decimal expansion of `p/q` using notation for repeating decimals.

Notation:

- Repeating part of the decimal is enclosed in parentheses.
- Example: `1/33 = 0.(03)`, `8639/70000 = 0.1234(142857)`

Hint: Use Long Division with Remainder Tracking

- Track each remainder seen during division.
- If a remainder repeats, the cycle starts again.
- Use a map from remainder to position to find the start of the repeating part.

Decimal Expansion of Rational Numbers

Example: 3/13

Step	Division	Remainder
1	$3/13 = 0$	$3\%13 = 3$
2	$30/13 = 2$	$30\%13 = 4$
3	$40/13 = 3$	$40\%13 = 1$
4	$10/13 = 0$	$10\%13 = 10$
5	$100/13 = 7$	$100\%13 = 9$
6	$90/13 = 6$	$90\%13 = 12$
7	$120/13 = 9$	$120\%13 = 3$
8	$30/13 = 2$	$30\%13 = 4$

Cycle starts repeating from Step 2.
Output: `0.(230769)`

Task: Write a C program:

- Reads 2 integers p and q from the user
- Prints the decimal expansion of p/q in the above format