# RAPTOR

RAPTOR (Rapid Algorithmic Prototyping Tool for Ordered Reasoning) is a flowchart-based programming environment that allows users to create algorithms using visual flowcharts rather than text-based code. RAPTOR is especially helpful for beginners, as it provides an intuitive way to design and visualize algorithms. It was developed to support learning in computer science and programming by making algorithmic thinking accessible and visual.

## Key Features of RAPTOR

- **Flowchart-Based Interface:** RAPTOR uses a flowchart-based interface, which makes it easy to visualize and understand the logical flow of an algorithm.
- **Supports Basic Control Structures:** RAPTOR supports essential programming constructs like loops, conditionals, and assignments.
- **Simple to Use:** The drag-and-drop interface and visual nature make it simple for beginners to use without needing prior coding experience.
- **Immediate Feedback:** RAPTOR allows for quick execution of algorithms, giving users immediate feedback on the output and allowing for debugging in real-time.
- **Error Detection:** RAPTOR identifies syntax errors and logical issues, which can help students learn and correct mistakes.

## Purpose of RAPTOR

The main purpose of RAPTOR is to help students and beginners understand programming concepts, logical flow, and problem-solving. It is widely used in education to teach programming logic before introducing syntax-heavy programming languages like Python or Java. By focusing on logic, RAPTOR provides a smooth learning curve for new learners.

## Components of the RAPTOR Interface

- **Start and End Symbols**: These symbols denote the beginning and end of a flowchart. Every RAPTOR flowchart starts with a "Start" symbol and ends with an "End" symbol.
- **Assignment Symbols**: Used to assign values to variables or to perform calculations. The symbol is represented by a parallelogram.
- **Input/Output Symbols**: RAPTOR uses parallelogram-shaped symbols for input and output operations. These allow the user to interact with the flowchart by taking inputs from the user and displaying outputs.
- **Decision Symbols**: Represented by diamond shapes, these symbols are used for conditional branching. A decision symbol has two paths, usually labeled "Yes" and "No," which enable the algorithm to follow different paths based on the condition.
- **Loop Symbols**: RAPTOR includes loop structures, allowing users to repeat parts of the flowchart multiple times. Looping helps manage repetitive tasks within algorithms.

## How to Use RAPTOR to Draw Flowcharts

### 1. Setting Up RAPTOR

1. Download and install the RAPTOR tool from the official website.
2. Open the RAPTOR application. You will be presented with a blank workspace where you can start creating flowcharts.

### 2. Creating a Basic Flowchart

1. **Start Symbol**: Begin by adding the "Start" symbol, which represents the beginning of the flowchart.
2. **Adding Input/Output Symbols**: To interact with the user, you can add input and output symbols.
   - Drag the "Input" symbol from the toolbar and connect it below the "Start" symbol. Double-click to specify the variable name (e.g., `Original_Price`) to receive input.

   - Add an "Output" symbol to display the results at the appropriate steps in the flowchart.
3. **Adding Assignments**: Use assignment symbols to perform calculations or assign values to variables.
   - For example, if you need to calculate the new price each day, drag an "Assignment" symbol and specify the calculation (e.g., `Price = Price * 0.95`).
4. **Using Decision Symbols for Conditional Logic**:
   - Drag the "Decision" symbol into the workspace if your algorithm includes conditions.
   - Enter the condition in the decision symbol (e.g., `Day < 7`). This will create two branches: "Yes" and "No," representing the possible outcomes.
   - Connect different paths based on the condition's outcome.
5. **Creating Loops**:
   - To repeat actions, use a loop by placing the "Loop" symbol.
   - Configure the loop with an appropriate counter variable (e.g., `Day = Day + 1`) and connect it back to the previous steps you want to repeat.
6. **Connecting Symbols**:
   - RAPTOR uses arrows to connect symbols and indicate the flow of execution. Connect symbols by dragging lines between them, forming a logical sequence from "Start" to "End."
   - Ensure all paths are correctly connected, especially for decision symbols, where there are two possible outcomes.
7. **End Symbol**: Conclude your flowchart with the "End" symbol to signify the termination of the algorithm.

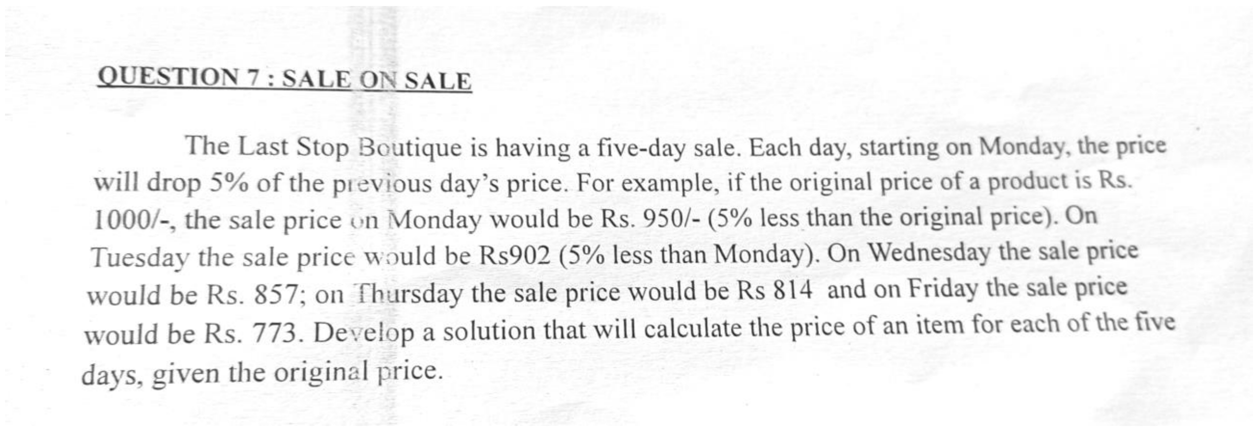
## 3. Running and Testing the Flowchart

1. **Execute the Flowchart**: RAPTOR allows you to run the flowchart to see the output. Click on the "Run" button to execute your flowchart.
2. **Debugging**: If there are any logical errors, RAPTOR will display error messages to help you debug. You can modify the flowchart as needed to correct any mistakes.
3. **Step-by-Step Execution**: RAPTOR provides an option to run the flowchart step-by-step, allowing you to observe each part of the process in detail.

This is especially helpful for understanding and learning how each symbol functions in the flow.

## 4. Saving and Exporting

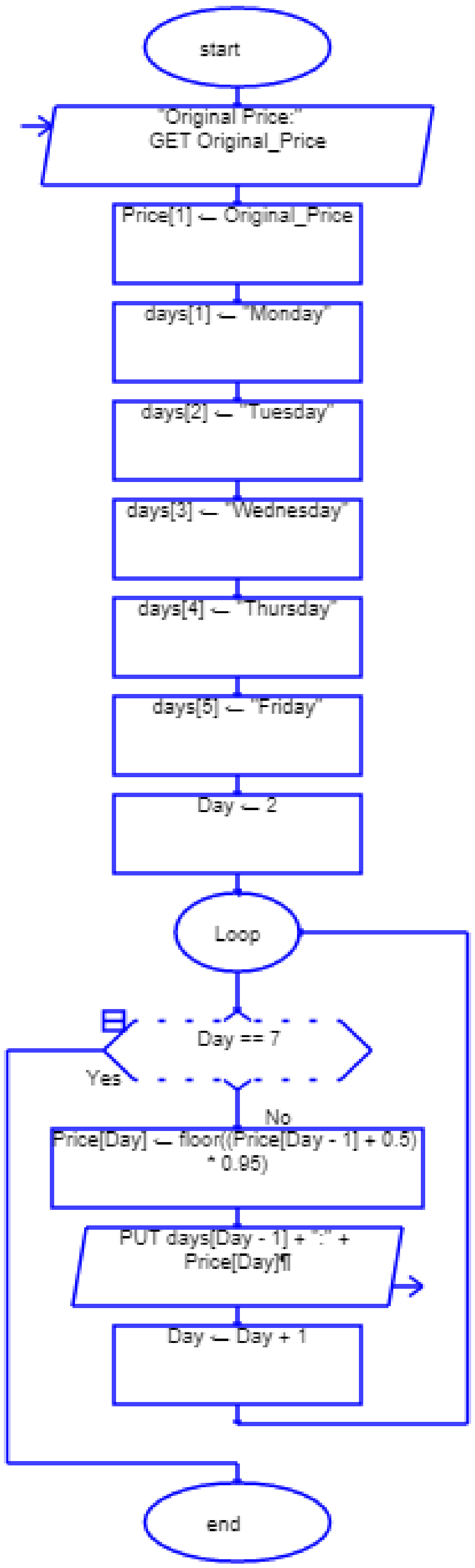
1. **Save**: You can save your flowchart for future use. This is helpful if you want to revisit the algorithm later or use it as a template.
2. **Export**: RAPTOR allows you to export flowcharts as images or files, which can be shared or embedded in documents.

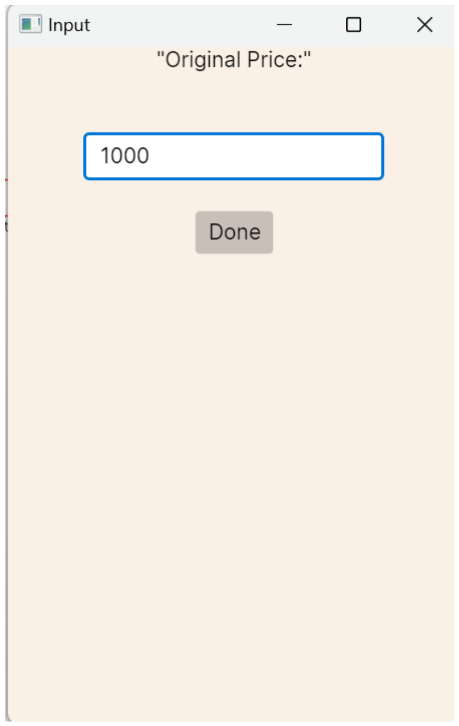


QUESTION 7 : SALE ON SALE

The Last Stop Boutique is having a five-day sale. Each day, starting on Monday, the price will drop 5% of the previous day's price. For example, if the original price of a product is Rs. 1000/-, the sale price on Monday would be Rs. 950/- (5% less than the original price). On Tuesday the sale price would be Rs902 (5% less than Monday). On Wednesday the sale price would be Rs. 857; on Thursday the sale price would be Rs 814 and on Friday the sale price would be Rs. 773. Develop a solution that will calculate the price of an item for each of the five days, given the original price.

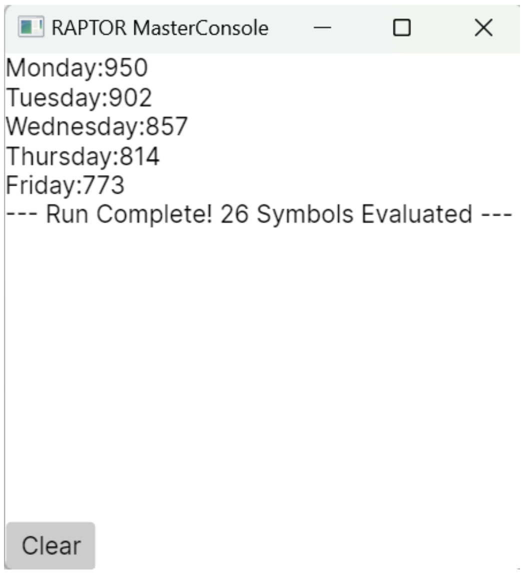
# RAPTOR FLOWCHART AND OUTPUT





"Original Price:"

1000



```
Monday:950
Tuesday:902
Wednesday:857
Thursday:814
Friday:773
--- Run Complete! 26 Symbols Evaluated ---
```

# ALGORITHM

**STEP 1:** START

**STEP 2:** Take the original price as input from the user and store it.

**STEP 3:** Create an array to store prices. Set the first element of this array to the original price.

**STEP 4**: Create an array with the names of the days: "Monday", "Tuesday", "Wednesday", "Thursday", and "Friday".

**STEP 5:** Start from the second day (representing Tuesday) and repeat the following steps until the fifth day (Friday):

- Add 0.5 to the previous day's price to help with rounding.
- Multiply the result by 0.95 to reduce the price by 5%.
- Round the result down to the nearest integer and store it as the current day’s price.
- Print the day’s name and the calculated price.
- Repeat this process until the prices for all five days (Monday to Friday) are calculated and printed

**STEP 6:** STOP

# PSEUDO CODE

```
1. Input Original_Price
2. Set Price[1] = Original_Price
3. Set days[1] = "Monday"
   Set days[2] = "Tuesday"
   Set days[3] = "Wednesday"
   Set days[4] = "Thursday"
   Set days[5] = "Friday"
4. For Day = 2 to 6 do:
     a. Set Price[Day] = floor((Price[Day - 1] + 0.5) * 0.95)
     b. Output days[Day - 2] + ": Rs " + Price[Day]
5. End
```

## PHOTO OF OUR GROUP DISCUSSING FOR THE PROJECT