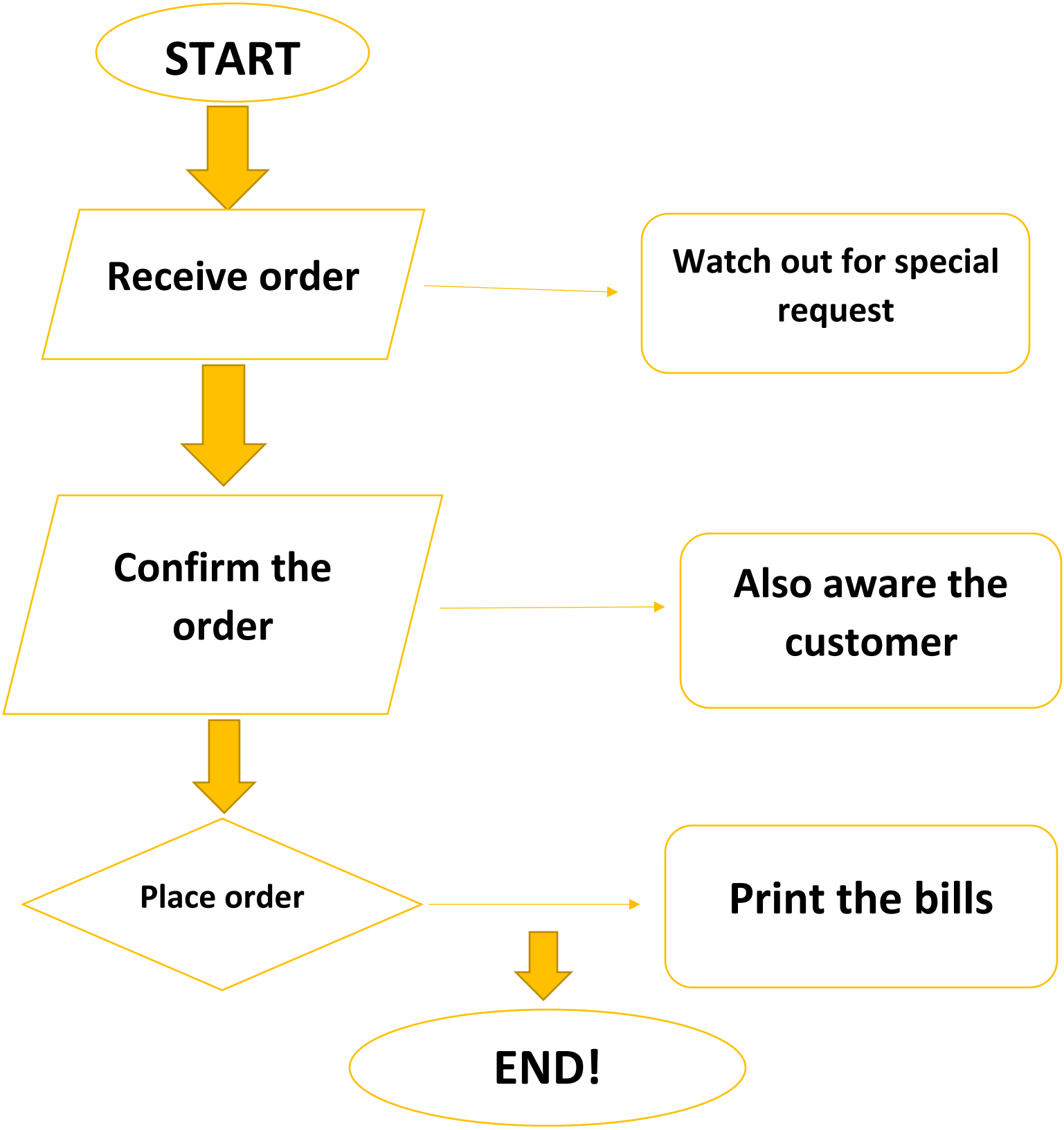
1.Design a flowchart, Pseudocode, Algorithm for processing a customer order at a restaurant, including handling special requests (Like add on).



Pseudo code

1. START
2. INPUT RECEIVE THE ORDER.
3. Confirm order if the customer wants the item available at that time
4. If item is not available then
5. Print (“sorry not available sir”)
6. If the item is available then
7. Print (“sure sir”)
8. Confirm order
9. Print bills
10. END!

ALGORITHM

1. Start
2. Place order of the customer
3. Receive the order
4. Confirm the order by also Alert the customer
5. Place order and print its bill
6. Make sure customer received the order
7. Ask for a feedback
8. END!

2.Design a flowchart, Pseudocode, Algorithm for handling a customer's deposit transaction at a bank, including checks for account validity and deposit amount conditions.

PSEUDO CODE

1. Start
2. Get Account NO
3. IF ACC: NO invalid
4. Print (“Invalid ACC: NO”)
5. RESET
6. Else proceed
7. Enter Amount NO:
8. If Amount: NO Invalid
9. Print (“Invalid Amount”)
10. RESET
11. Else proceed
12. Withdraw the cash
13. END!

ALGORITHM

1. **Start the transaction process.**
2. **Input the account number from the customer.**
3. **Verify if the provided account number is valid:**

4. **If invalid:**

    - Display an error message: "Invalid Account."
    - End the process

5. **If valid:**

    - Proceed to the next step.

6. **Input the deposit amount from the customer.**
7. **Check if the deposit amount is greater than 0:**

9. **If the amount is less than or equal to 0:**
    a. Display an error message: "Invalid Deposit Amount."
    b. End the process.
10. **If the amount is valid:**
    a. Proceed to the next step.

11. **Update the account balance by adding the deposit amount.**
12. **Display a success message: "Deposit Successful."**
13. **End the transaction process.**

3.Design a flowchart, Pseudocode, Algorithm to determine which of three provided numbers is the greatest.

PSEUDO CODE

```
1. BEGIN
2. // Input three numbers
   INPUT A, B, C

3. // Compare A and B
   IF A > B THEN
4.    // Compare A with C
      IF A > C THEN
         OUTPUT A
      ELSE
         OUTPUT C
      ENDIF
   ELSE
5.    // Compare B with C
      IF B > C THEN
         OUTPUT B
      ELSE
         OUTPUT C
      ENDIF
   ENDIF
END
```

ALGORITHM

1. **Start**
2. **Input** three numbers: `A`, `B`, `C`.
3. **If** `A > B`, **then**

    - **If** `A > C`, **then** `A` is the greatest.
    - **Else** `C` is the greatest.

3. **Else**, **if** `B > C`, **then** `B` is the greatest.

    - **Else** `C` is the greatest.

4. **Output** the greatest number.
5. **End**

4.Implement an algorithm and pseudocode where the user enters a number, and an appropriate month is displayed.

PSEUDOCODE

0. START
1. PROMPT "Enter a number between 1 and 12: "
2. READ number

3. IF number == 1 THEN
4. DISPLAY "January"

5. ELSE IF number == 2 THEN
6. DISPLAY "February"
7. ELSE IF number == 3 THEN
8. DISPLAY "March"
9. ELSE IF number == 4 THEN
10. DISPLAY "April"
11. ELSE IF number == 5 THEN
12. DISPLAY "May"
13. ELSE IF number == 6 THEN
14. DISPLAY "June"
15. ELSE IF number == 7 THEN
16. DISPLAY "July"
17. ELSE IF number == 8 THEN
18. DISPLAY "August"
19. ELSE IF number == 9 THEN
20. DISPLAY "September"
21. ELSE IF number == 10 THEN
22. DISPLAY "October"
23. ELSE IF number == 11 THEN
24. DISPLAY "November"
25. ELSE IF number == 12 THEN
26. DISPLAY "December"
27. ELSE DISPLAY "Error: Please enter a number between 1 and 12."
28. END IF
29. END

ALGORITHM

1. Start.
2. Prompt the user to enter a number between 1 and 12.
3. Read the user's input.

4. Use a selection structure (if-else or switch-case) to match the number with the corresponding month.
5. Display the name of the month based on the input number.
6. If the number is not between 1 and 12, display an error message.
7. End.

5. Create pseudocode a small calculator which only does ‘+’ or ‘-’Operations. (Hint: Take three variable inputs with one being used for the operator)

Algorithm

1. START.
2. Prompt the user to enter the first number.
3. Read the first number and store it in number1.
4. Prompt the user to enter the operator (+ or -).
5. Read the operator and store it in operator.
6. Prompt the user to enter the second number.
7. Read the second number and store it in number2.
8. Check the value of operator:

- If the operator is +, calculate the sum of number1 and number2, and store the result in result.
- If the operator is -, calculate the difference between number1 and number2, and store the result in result.

9. Display the value of result.

10. If the operator is neither + nor -, display an error message indicating an invalid operator.
11. End.

Pseudocode

1. BEGIN
2. PROMPT "Enter the first number: "
3. READ number1
4. PROMPT "Enter the operator (+ or -): "
5. READ operator PROMPT "Enter the second number: "
6. READ number2 IF operator == '+' THEN result = number1 + number2
7. DISPLAY "Result: " + result
8. ELSE IF operator == '-'
9. THEN result = number1 - number2
10. DISPLAY "Result: " + result
11. ELSE
12. DISPLAY "Error: Invalid operator.
    Please use either + or -."

END IF

13. END

6.You are working at Toyota Indus Motors and want to assemble a car. make algorithm and pseudocode replicate a pipeline production.

ALGORITHM

1. **Start**.
2. **Frame Assembly**:

- Prepare and weld the car's frame.
- Check for any defects or misalignments in the frame.

3. **Engine Installation**:

- Install the engine into the frame.
- Secure the engine with bolts and other necessary components.
- Perform a basic engine test to ensure correct installation.

4. **Transmission Installation**:

- Install the transmission system.
- Connect the transmission to the engine.
- Verify that the transmission operates smoothly.

5. **Suspension and Wheels Installation**:

- Attach the suspension system to the frame.
- Install wheels on all four corners.
- Ensure all wheels are properly aligned.

6. **Interior Assembly**:

- Install the dashboard, seats, and other interior components.
- Check for proper fitting and finish of all interior parts.

7. **Electrical Systems Installation**:

- Connect the car's electrical systems (lights, sensors, control systems).
- Test all electrical components to ensure they work properly.

8. **Body Painting and Finishing**:

- Paint the car's exterior according to the design.
- Apply any decals or final touches.
- Inspect the paint job for any imperfections.

9. **Final Inspection and Testing**:

- Conduct a thorough inspection of the assembled car.
- Perform a test drive to ensure all systems are working correctly.
- Address any issues found during the inspection and testing.

10. **Packaging and Delivery**:

- Prepare the car for delivery.
- Document the car's completion and schedule delivery.

11. **End**.

PSEUDOCODE

1. **Frame Assembly:** Start by assembling the frame, ensuring it's constructed properly. Once done, inspect it thoroughly for any issues.
2. **Engine Installation:** Next, install the engine securely. After installation, run a test to confirm it's installed correctly.
3. **Transmission Installation:** Move on to installing the transmission. Check to make sure everything is functioning as expected after it's in place.
4. **Suspension and Wheels:** Install the suspension system, then attach the wheels. Align the wheels for proper balance and handling.

5. **Interior Assembly:** Begin installing the dashboard and seats. Once installed, double-check that everything fits well and is secure.
6. **Electrical Systems Installation:** Install all necessary electrical systems, then conduct a test to verify they're working properly.
7. **Body Painting and Finishing:** Paint the car in the desired color, apply decals or additional styling elements, and inspect the paint job to ensure it's flawless.
8. **Final Inspection and Testing:** Perform a thorough final inspection. Conduct a test drive to ensure everything is in working order, and address any issues that arise.
9. **Packaging and Delivery:** Prepare the vehicle for delivery, document the completion of the process, and schedule the delivery to the customer.

7. Implement an algorithm and pseudocode for making a simple calculator with all the operators (+,-,*,/,%)

ALGORITHM

1. **Start**
2. **Input**: Prompt the user to enter the first number.
3. **Input**: Prompt the user to enter the operator (+, -, *, /, %).
4. **Input**: Prompt the user to enter the second number.
5. **Operation**:

- If the operator is +, calculate the sum of the two numbers.
- If the operator is -, calculate the difference between the two numbers.
- If the operator is *, calculate the product of the two numbers.
- If the operator is /, check if the second number is not zero, then calculate the quotient.
- If the operator is %, check if the second number is not zero, then calculate the remainder.

6. **Output**: Display the result of the operation.
7. **Error Handling**:

- If the operator is not one of the specified symbols (+, -, *, /, %), display an error message.
- If division or modulus by zero is attempted, display an error message indicating that division by zero is not allowed.

8. **End**.

PSEUDOCODE

1. Start by asking the user to input the first number:
2. Enter the first number:
3. Store this number in **number1**.
4. Next, prompt the user to choose an operator:
5. Enter the operator (+, -, *, /, %):
6. Store the chosen operator in **operator**.
7. Then, ask the user to input the second number:
8. Enter the second number:

9. Store this number in **number2**.
10. Now, proceed with the operation:

- If the operator is **+**, calculate the sum of **number1** and **number2**, then display the result.
- If the operator is **-**, subtract **number2** from **number1**, then display the result.
- If the operator is *****, multiply **number1** by **number2**, then display the result.
- If the operator is **/**, first check if **number2** is not zero. If it's not, divide **number1** by **number2**, then display the result. If **number2** is zero, display an error message indicating that division by zero is not allowed.
- If the operator is **%**, check again if **number2** is not zero. If it's not, find the remainder of **number1** divided by **number2**, then display the result. If **number2** is zero, display an error message indicating that division by zero is not allowed.
- If the operator doesn't match any of the allowed symbols (**+**, **-**, *****, **/**, **%**), display an error message indicating that an invalid operator was entered.

11. End the program.

9.Why we use Git-ignore?

A Git-ignore file is used to tell Git which files or folders it should ignore and not track.

10.Difference between Algorithm and Pseudocode?

An algorithm is essentially a step-by-step plan for solving a problem. Pseudocode, on the other hand, is a method for expressing that plan in a format that resembles actual code but remains easy to understand.

THE % END