**1. Design a flowchart, Pseudocode, Algorithm for processing a customer order at a restaurant,**

**including handling special requests (Like add on).**

Print:

Enter your Order:

If ordered item is available

**YES NO**

Print:

Do you want to order more?

Add to the order list

Print:

Ordered Item is not available

**YES NO**

**Pseudocode:**

1. **PRINT “ENTER YOUR ORDER”**
2. **INPUT ORDER**
3. **SET LIST TO NILL**
4. **IF ORDER IS AVAILABE THEN**

**ADD ORDER TO LIST**

1. **ELSE**

**PRINT “ORDERED ITEM IS NOT AVAILABLE”**

1. **PRINT “DO YOU WANT TO ORDER MORE?”**
2. **IF YES**

**INPUT ORDER**

**ADD ORDER TO LIST**

1. **ELSE**

**END**

**Algorithm:**

* **Ask the user to enter order**
* **Check whether the ordered item is present or not:**

**If yes, add the ordered item to the list**

**If no, Print “Ordered item not available”**

* **Ask user if they want to order more**

**If yes, ask user to enter their order and add it in the list.**

**If no, then close the list.**

**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.**

**Pseudocode:**

1. **INPUT PIN**
2. **IF PIN==CORRECT**

**(**

**INPUT AMOUNT**

**IF AMOUNT IS AVAILABLE**

**CASH OUT THE AMOUNT**

**ELSE**

**(**

**PRINT” THAT MUCH AMOUNT IS NOT PRESENT IN YOUR ACCOUNT”**

**END**

**)**

**)**

1. **ELSE**

**END**

**Algorithm:**

* **Ask user his account pin**
* **If pin is correct, ask user the amount of money to cash out.**

**If that amount of money is present in users account, then cash out the amount.**

**Else print “THAT MUSCH AMOUNT OF MONEY IS NOT PRESENT IN YOUR ACCOUNT”.**

**And now end.**

* **If pin is incorrect then print “Pin is incorrect”**
* **End the task**

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

**greatest.**

**Pseudocode:**

1. **INPUT NUMBER1**
2. **INPUT NUMBER2**
3. **INPUT NUMBER3**
4. **IF (NUMBER1>NUMBER2) && (NUMBER1>NUMBER3)**

**PRINT NUMBER1**

**ELSE IF (NUMBER2>NUMBER1) && (NUMBER2>NUMBER3)**

**PRINT NUMBER2**

**ELSE**

**PRINT NUMBER3**

1. **END**

**Algorithm:**

* **Ask user to input 3 random number and store them in NUMBER1, NUMBER2 and NUMBER3 respectively.**
* **If NUMBER1 is greater than NUMBER2 and NUMBER3 then print NUMBER1.**
* **Else if NUMBER2 is greater than NUMBER1 and NUMBER3 then print NUMBER2.**
* **Else print NUMBER3.**

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

**displayed.**

**Pseudocode:**

1. **INPUT NUM**
2. **SWITCH (NUM)**

**CASE 1:**

**PRINT “JANUARY”**

**CASE 2:**

**PRINT “FEBURARY”**

**CASE 3:**

**PRINT “MARCH”**

**CASE 4:**

**PRINT “APRAIL”**

**CASE 5:**

**PRINT “MAY”**

**CASE 6:**

**PRINT “JUNE”**

**CASE 7:**

**PRINT “JULY”**

**CASE 8:**

**PRINT “AUGUST”**

**CASE 9:**

**PRINT “SEPTEMBER”**

**CASE 10:**

**PRINT “OCTOMBER”**

**CASE 11:**

**PRINT “NOVEMBER”**

**CASE 12:**

**PRINT “DECEMBER”**

1. **END SWITCH**
2. **END**

**Algorithm:**

* **Ask user to enter any number between 1 and 12 and store it in NUM.**
* **Find the month name corresponding to that number using switch case.**
* **Print month name.**

**5. Create pseudocode a small calculator which only does ‘+’ or ‘- ‘Operations. (Hint: Take three**

**variable inputs with one being used for the operator)**

**Pseudocode:**

1. **INPUT OPERATOR**
2. **INPUT NUM1**
3. **INPUT NUM2**
4. **SET ANS TO 0**
5. **IF OP==’+’**

**ANS=NUM1+NUM2**

**ELSE**

**ANS=NUM1-NUM2**

1. **PRINT ANS**
2. **END**

**Algorithm:**

* **Ask user to input two random numbers and an operator (+ or -) and store them in NUM1, NUM2 and OPERATOR respectively.**
* **Take the NUM1 and NUM2 and perform the given operation between the two numbers.**
* **Store the answer in ANS.**
* **Print ANS.**

**6. You are working at Toyota Indus Motors and want to assemble a car. Design a flowchart with**

**proper process modules and decision structures to replicate a pipeline production.**

**Pseudocode:**

1. **START**
2. **GET ORDER**
3. **DESIGN FRAME OF CAR**
4. **ADD WHEELS**
5. **INSTALL ENGINE**
6. **IF ENGINE IS WORKING PROPERLY**

**PROCCED**

**ELSE**

**RECHECK THE ENGINE**

1. **INSTALL WIRING**

**IF WIRING IS WORKING PROPERLY**

**PROCCED**

**ELSE**

**RECHECK THE WIRING**

1. **ATTACH SEATS**
2. **CREATE INTERIOR**
3. **PAINT THE CAR**
4. **DELIVER THE CAR**
5. **END**

**Algorithm:**

* Receive an order for a car.
* Design the car frame according to specifications.
* Attach wheels to the designed frame.
* Install the engine in the car frame.
* If the engine is working properly, proceed to step 7.
* Else, recheck and fix the engine.
* Install the electrical wiring in the car.
* If the wiring is working properly, proceed to step 9.
* Else, recheck and fix the wiring.
* Install the seats inside the car.
* Complete the interior design by adding necessary components like dashboard, upholstery, etc.
* Paint the car according to the specified color and finish.
* Prepare the car for delivery to the customer.
* End

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

**Pseudocode:**

1. **INPUT OPERATOR**
2. **INPUT NUM1**
3. **INPUT NUM2**
4. **SET ANS TO 0**
5. **IF OP==’+’**

**ANS=NUM1+NUM2**

**ELSE IF OP==’-’**

**ANS=NUM1-NUM2**

1. **ELSE IF OP==’\*’**

**ANS=NUM1\*NUM2**

1. **ELSE IF OP==’/’**

**ANS=NUM1/NUM2**

1. **ELSE**

**ANS=NUM1%UM2**

1. **PRINT ANS**
2. **END**

**Algorithm:**

* **Ask user to input two random numbers and an operator (+ or – or / or \* or %) and store them in NUM1, NUM2 and OPERATOR respectively.**
* **Take the NUM1 and NUM2 and perform the given operation between the two numbers.**
* **Store the answer in ANS.**
* **Print ANS.**

**9. Why we use .gitignore?**

**We use .gitignore to ignore the files and directories by Git when we add or commit files to a repository. It helps Git ignore the files like temporary files created by text editor, which helps to track the repository faster and makes the repository cleaner. By using .gitignore, we make the Git to ignore the files which contains sensitive data like API Key, passwords etc. so that they don’t get deleted by mistake.**

**10. Difference between Algorithm and Pseudocode?**

**Algorithm: Detailed method for solving a problem; more formal and close to final implementation.**

**Pseudocode: Readable representation of an algorithm’s logic; bridges the gap between flowcharts and code.**

**Algorithm: Abstract and mathematical; can be in natural language or mathematical notation.**

**Pseudocode: Mixes natural language with code-like structure; avoids specific syntax rules.**

**Algorithm: For analysis by computer scientists, mathematicians, and developers.**

**Pseudocode: For programmers and educators to understand and explain algorithm logic before coding.**

**Algorithm: High-level description; doesn’t include specific programming details.**

**Pseudocode: More detail on implementation steps; doesn’t get bogged down in syntax.**