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

including handling special requests (Like add on).

**FLOWCHART:**

Start

Process

Add the Things to the bill

Input

Enter The Things you want to buy

Would you like to add any thing

Yes

No

Output

Print The Bill

End

**ALGORITHM:**

1. Start

2. Receive the customer's order.

3. Add to the bill

4. Would You Like to add any thing if yes go to step 2 else continue

5. Print the Bill

6. Once the order is prepared by the kitchen: a. Deliver the order to the customer.

7. End

**Pseudocode:**

1. Start

2. Receive Order items

3. Add items to the Bill

3.If customer want to add items Then

Go to step 2

Else

Go to next step

4. print the bill

5. End

Q. Design a flowchart, Pseudocode, Algorithm for handling a customer's deposit transaction at a

bank, including checks for account validity and deposit amount conditions.

**FLOWCHART:**

Start

Receive Deposit Request

Validate Account

invalid

Output

Transaction Rejected

valid

Validate Deposit Amount

valid

End

Output

Transaction successful

Update Account Balance

Process Deposit

**ALGORITHM:**

1. Start

2. Receive the deposit request from the customer, including account details and deposit amount.

3. Validate the account:

a. If the account is valid:

i. Validate the deposit amount:

- If the deposit amount is valid (e.g., positive amount):

a. Process the deposit.

b. Update the account balance.

c. Confirm the transaction success to the customer.

- Else (if the deposit amount is invalid):

a. Reject the transaction.

b. Else (if the account is invalid):

i. Reject the transaction.

4. End

**Pseudocode:**

Start

Receive Deposit Request

If Account is Valid Then

If Deposit Amount is Valid Then

Process Deposit

Update Account Balance

Confirm Transaction Success

Else

Reject Transaction

Else

Reject Transaction

End

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

greatest.

**FLOWCHART:**

Start

End

Print Largest

else

Largest = Num3

Else If Num2>Num1 &&

Num2>Num3

Then Largest = Num2

If Num1>Num2 &&

Num1>Num3

Then Largest = Num1

Input

Num1, Num2, Num3

**ALGORITHM:**

1. Start
2. Input Num1, Num2, Num3
3. If num1 > Num2 && Num1 > Num3 then largest=num1
4. Else if num2 > Num1 && Num2 > Num3 then largest=num2
5. Else largest=num3
6. Print largest
7. End

**Pseudocode:**

Start

Input num1, num2, num3

If num1 > num2 and num1 > num3

Largest = num1

Else if num2 > num1 and num2 > num3

Largest = num2

Else

Largest = num3

Print largest

End

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

displayed.

1. Start

2. Input a number (MonthNumber) from the user.

3. Check if the number is within the valid range (1 to 12):

a. If MonthNumber is between 1 and 12:

i. Use a switch or conditional statement to map MonthNumber to its corresponding month:

- If MonthNumber is 1, display "January".

- If MonthNumber is 2, display "February".

- If MonthNumber is 3, display "March".

- If MonthNumber is 4, display "April".

- If MonthNumber is 5, display "May".

- If MonthNumber is 6, display "June".

- If MonthNumber is 7, display "July".

- If MonthNumber is 8, display "August".

- If MonthNumber is 9, display "September".

- If MonthNumber is 10, display "October".

- If MonthNumber is 11, display "November".

- If MonthNumber is 12, display "December".

b. Else (MonthNumber is outside the range 1 to 12):

i. Display "Invalid number. Please enter a number between 1 and 12."

4. End

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

variable inputs with one being used for the operator)

Start

// Input the two numbers and the operator

Input Number1

Input Number2

Input Operator

// Perform operation based on the operator

If Operator is '+' Then

Result = Number1 + Number2

Output "The result of", Number1, "+", Number2, "is", Result

Else If Operator is '-' Then

Result = Number1 - Number2

Output "The result of", Number1, "-", Number2, "is", Result

Else

Output "Invalid operator. Please enter '+' for addition or '-' for subtraction."

End

Q. 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.

start

Receive Order

Prepare assembly lines

Inspect components

ALL OK

Not ok

Reorder Components

Assemble Car

Quality Check

pass

Fail

Rework on car

End

Deliver Car to customer

Finalize car

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

1. Start

2. Input Operand1

3. Input Operand2

4. Input Operator

5. If Operator is '+'

a. Result = Operand1 + Operand2

6. Else If Operator is '-'

a. Result = Operand1 - Operand2

7. Else If Operator is '\*'

a. Result = Operand1 \* Operand2

8. Else If Operator is '/'

a. If Operand2 is not zero

i. Result = Operand1 / Operand2

b. Else

i. Output "Error: Division by zero"

9. Else If Operator is '%'

a. Result = Operand1 % Operand2

10. Else

a. Output "Invalid operator"

11. Output Result

12. End

Q. Why we use .gitignore?

The `.gitignore` file is essential for managing a Git repository by specifying which files and directories should be excluded from version control. It helps prevent unnecessary files, like temporary build artifacts or sensitive configuration files, from being tracked, thereby keeping the repository clean and efficient. By using `.gitignore`, developers avoid cluttering the commit history with irrelevant changes, protect sensitive information, and ensure a consistent development environment across all collaborators. This results in a more organized and manageable codebase.

Q. Difference between Algorithm and Pseudocode?

**Algorithm**

**Definition:** An algorithm is a step-by-step procedure or set of rules for solving a problem or performing a task. It’s a high-level description of how to achieve a specific goal or outcome.

### Pseudocode

**Definition:** Pseudocode is a method of designing algorithms using a structured but informal language that resembles programming constructs but is not tied to any specific programming syntax. It’s used to describe algorithms in a way that is easy to understand and can be translated into actual code later.