1. Student Grade Calculator

Program Description: Create a program that calculates the final grade for a student based on assignments (30%), midterm exam (30%), and final exam (40%). The program should determine if the student passed (≥60%) or failed.

Key Features:

Input assignment scores, midterm score, and final exam score

Calculate weighted average based on predefined percentages

Determine pass/fail status

Display final grade and status to user

**Algorithm:**

Step 1: Start

Step 2: Input assignments, midterm\_marks, final\_marks

Step 3: Calculate final grade using weighted average

Final\_marks = (assignments\*0.30)+ (midterm\_marks\*0.30)+ (final\_marks\*0.40)

Step 4: if final\_marks>=60 then status= “pass”

Else status= ”fail”;

Step 5: Print “final grade:”, final\_grade,%

Print “status:”, status

Step 6: Stop

**Pseudocode:**

Step 1: START

Step 2: INPUT assignment\_marks, midterm\_marks, finalexam\_marks

Step 3: final\_grade = (assignment\_marks\*0.30) + (midterm\_marks\*0.30) + (finalexam\_marks\*0.40)

Step 4: IF final\_grade >= 60 THEN status = "pass"

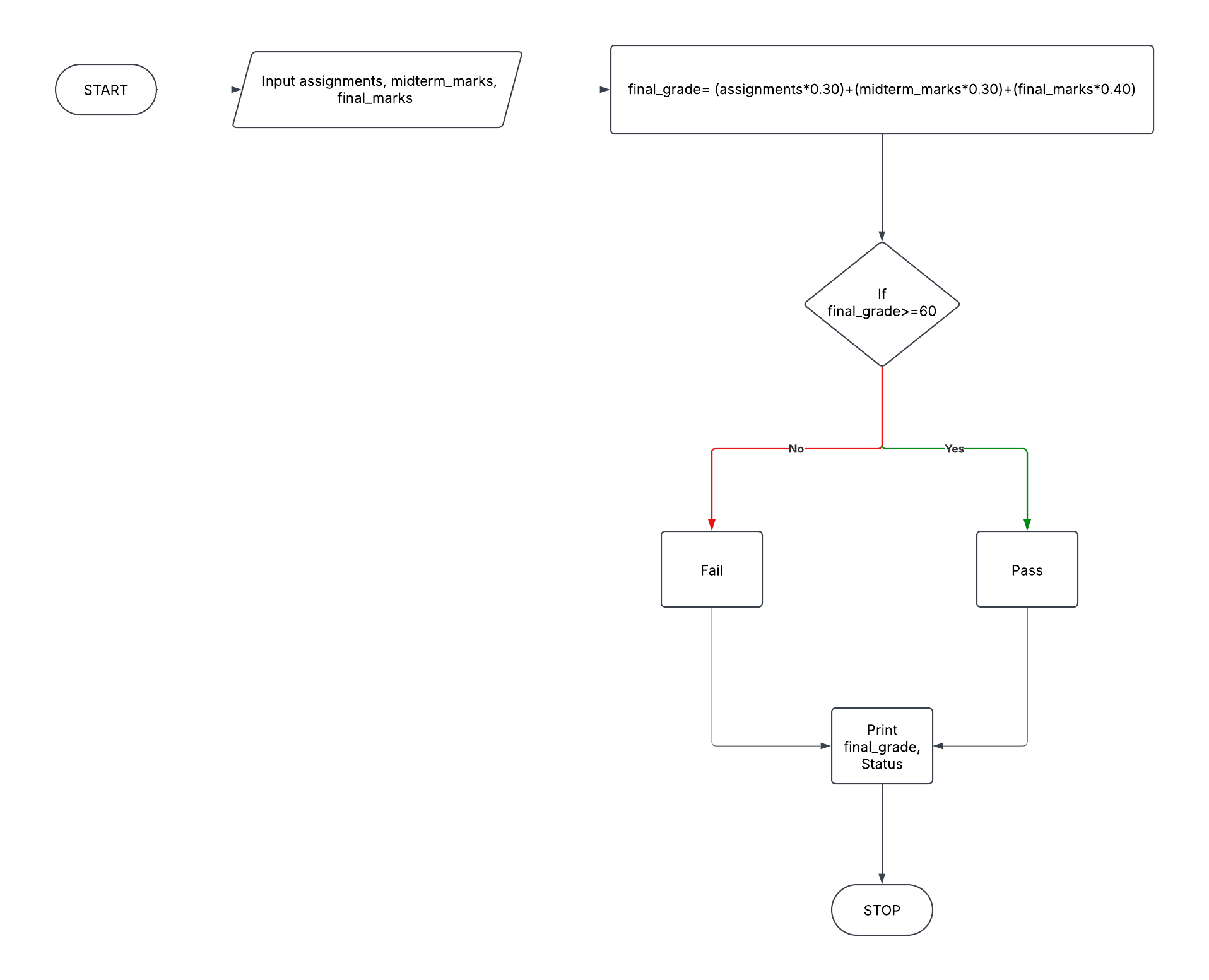
Else status="fail"

Step 5: PRINT "Final grade:", final\_grade,"%"

PRINT "status", status

Step 6: STOP

**FlowChart:**



**2. ATM Banking System**

**Program Description: Develop a program that simulates an ATM with options to check balance, deposit money, withdraw money, and exit. The program should maintain a running balance and prevent withdrawals that would result in a negative balance.**

**Key Features:**

* **Authenticate user with PIN**
* **Display menu of available operations**
* **Handle balance inquiries**
* **Process deposits and update balance**
* **Validate withdrawal requests against available balance**
* **Provide transaction receipts**
* **Allow user to exit system**

**Algorithm:**

Step 1: Start

Step 2: Set initial balance

Step 3: Authenticate user:

Get user to enter pin

If pin is incorrect, allow up to 3 attempts before exiting

Step 4: Display menu:  
 1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Exit

Step 5: Process user selection:

If option=1 :> Display balance

If option=2 :>

Prompt user to enter deposit amount

Add amount to balance

Display new balance

If option=3 :>

Prompt user to enter withdrawl amount

If amount>balance -> Display “Insuffient funds”

Else, deduct amount from balance and display new balance

If option=4 :>

Exit the system

Step 6: Repeat until the user exists

Step 7: Stop

**Pseudocode:**

STEP 1: START

Step 2: SET balance=1000

SET correct\_pin= 1234

SET attempts = 0

SET max\_attempts = 3

Step 3: REPEAT

PRINT “Enter your pin:”

INPUT entered\_pin

IF entered\_pin == correct\_pin THEN

BREAK

ELSE

INCREMENT attempts

PRINT “Incorrect\_pin. Try again”

UNTIL attempts == max\_attempts

IF attempts == max\_attempts THEN

PRINT “Too many failed attempts. Exiting”

STOP

ENDIF

Step 4: REPEAT

PRINT “ATM Menu:”

PRINT “1. Check Balance”

PRINT “2. Deposit Money”

PRINT “3. Withdraw Money”

PRINT “4. Exit”

PRINT “Enter your choice:”

INPUT choice

Step 5: Process user choice

IF choice == 1 THEN

PRINT “Your balance is:”, balance

ELSE IF choice ==2 THEN

PRINT “Enter deposit amount:”

INPUT deposit

Balance = balance+deposit

PRINT “New balance:”, balance

ELSE IF choice==3 THEN

PRINT “Enter withdrawl amount:”

INPUT withdrawl

IF withdrawl > balance THEN

PRINT “Insufficient funds.”

ELSE

balance = balance-withdrawl

PRINT “New balance:”, balance

END IF

ELSE

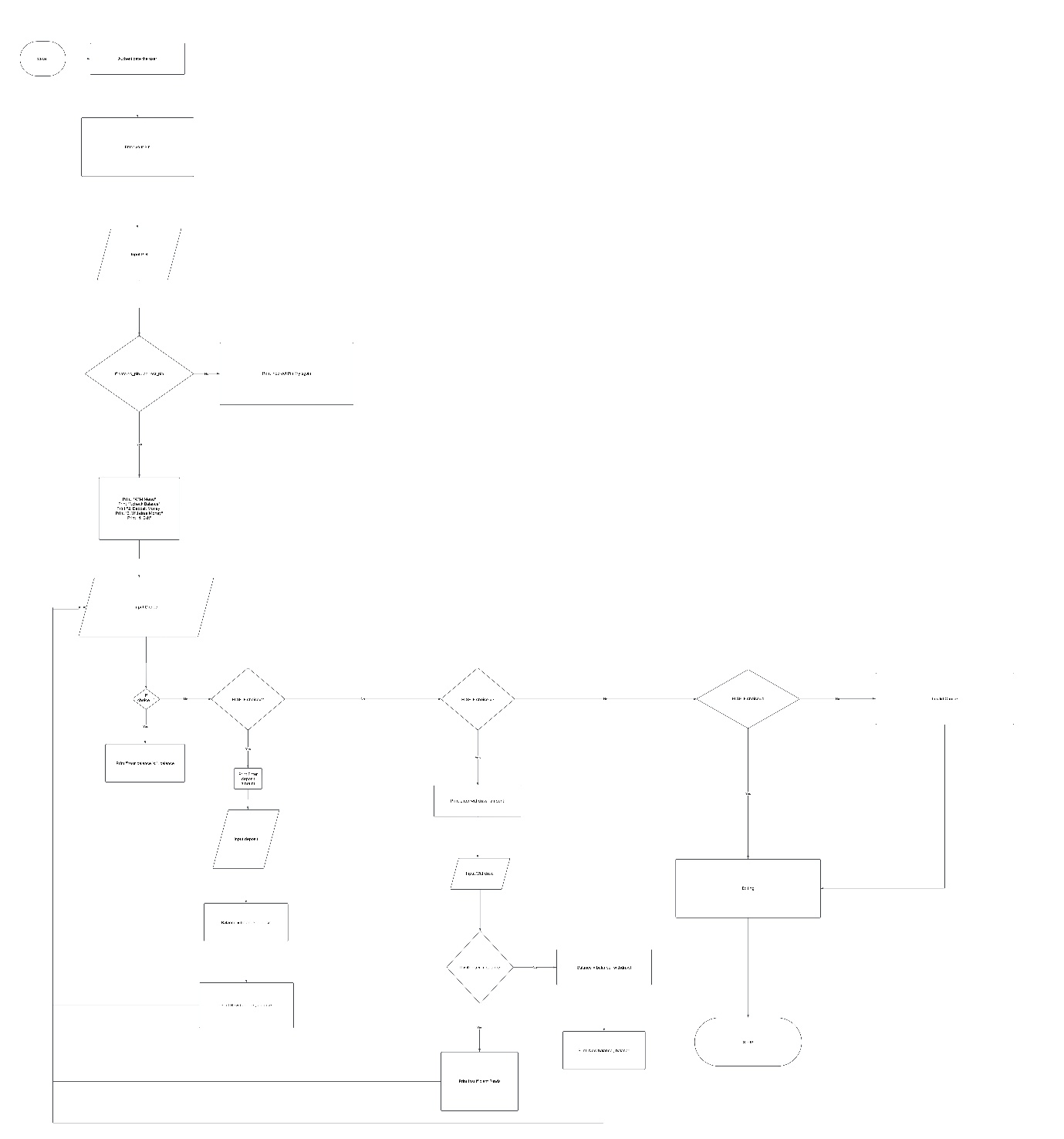
PRINT “Invalid option. Try again”

ENDIF

UNTIL choice == 4

Step 6: STOP

**FLOWCHART:**

****

**3. Inventory Management System**

**Program Description: Design a program that manages a store's inventory by allowing users to add new items, update quantities, remove items, and display the current inventory. Each item should have an ID, name, price, and quantity.**

**Key Features:**

* **Add new products to inventory with unique IDs**
* **Update existing product information**
* **Remove products from inventory**
* **Search for products by ID or name**
* **Display current inventory status**
* **Track low stock items**
* **Generate inventory reports**

**Algorithm:**

**Step 1 :** Start

Step 2: Initialize an empty inventory list

Step 3: Display Menu options

1. Add a new product

2. Update product quality

3. Remove a product

4. Search for a product

5. Display all products

6. View low stock items

7. Generate Inventory report

8. Exit

Step 4: Process user selection

If option=1

Input product ID, name, price and quality

Ensure ID is unique

Add product to inventory

If option=2

Input product ID

If found, update the quality

If not found, display an error

If option=3

Input product ID

Id found, remove producr from inventory

If option=4

Input ID or name

Display product details if found

If option=5

Print all products in the inventory

If option=6

Show products where quantity < threshold

If option = 7

Display all inventory details in a structured format

If option = 8

End the program

Step 5: Repeat until the user selects Exit

Step 6: Stop

**Pseudocode:**

Step 1: START

Step 2: DECLARE inventory as an empty list

Step 3: REPEAT

PRINT "Inventory Management System Menu"

PRINT "1. Add Product"

PRINT "2. Update Product Quantity"

PRINT "3. Remove Product"

PRINT "4. Search Product"

PRINT "5. Display All Products"

PRINT "6. View Low Stock Items"

PRINT "7. Generate Inventory Report"

PRINT "8. Exit"

PRINT "Enter your choice:"

INPUT choice

IF choice == 1 THEN

PRINT "Enter Product ID: "

INPUT product\_ID

PRINT "Enter Product Name: "

INPUT product\_name

PRINT "Enter Product Price: "

INPUT product\_price

PRINT "Enter Product Quantity: "

INPUT product\_quantity

ADD (product\_ID, product\_name, product\_price, product\_quantity) TO inventory

ELSE IF choice == 2 THEN

PRINT "Enter Product ID to update: "

INPUT product\_ID

IF product\_ID EXISTS in inventory THEN

PRINT "Enter new quantity: "

INPUT new\_quantity

UPDATE inventory[product\_ID].quantity = new\_quantity

ELSE

PRINT "Product not found!"

ENDIF

ELSE IF choice == 3 THEN

PRINT "Enter Product ID to remove: "

INPUT product\_ID

IF product\_ID EXISTS in inventory THEN

REMOVE product from inventory

ELSE

PRINT "Product not found!"

ENDIF

ELSE IF choice == 4 THEN

PRINT "Enter Product ID or Name to search: "

INPUT search\_term

IF search\_term EXISTS in inventory THEN

DISPLAY product details

ELSE

PRINT "Product not found!"

ENDIF

ELSE IF choice == 5 THEN

DISPLAY all products in inventory

ELSE IF choice == 6 THEN

PRINT "Low Stock Products (Less than 5 items)"

FOR each product in inventory DO

IF product.quantity < 5 THEN

DISPLAY product details

ENDIF

ENDFOR

ELSE IF choice == 7 THEN

PRINT "Inventory Report:"

DISPLAY all product details in a structured format

ELSE IF choice == 8 THEN

PRINT "Exiting System..."

BREAK

ELSE

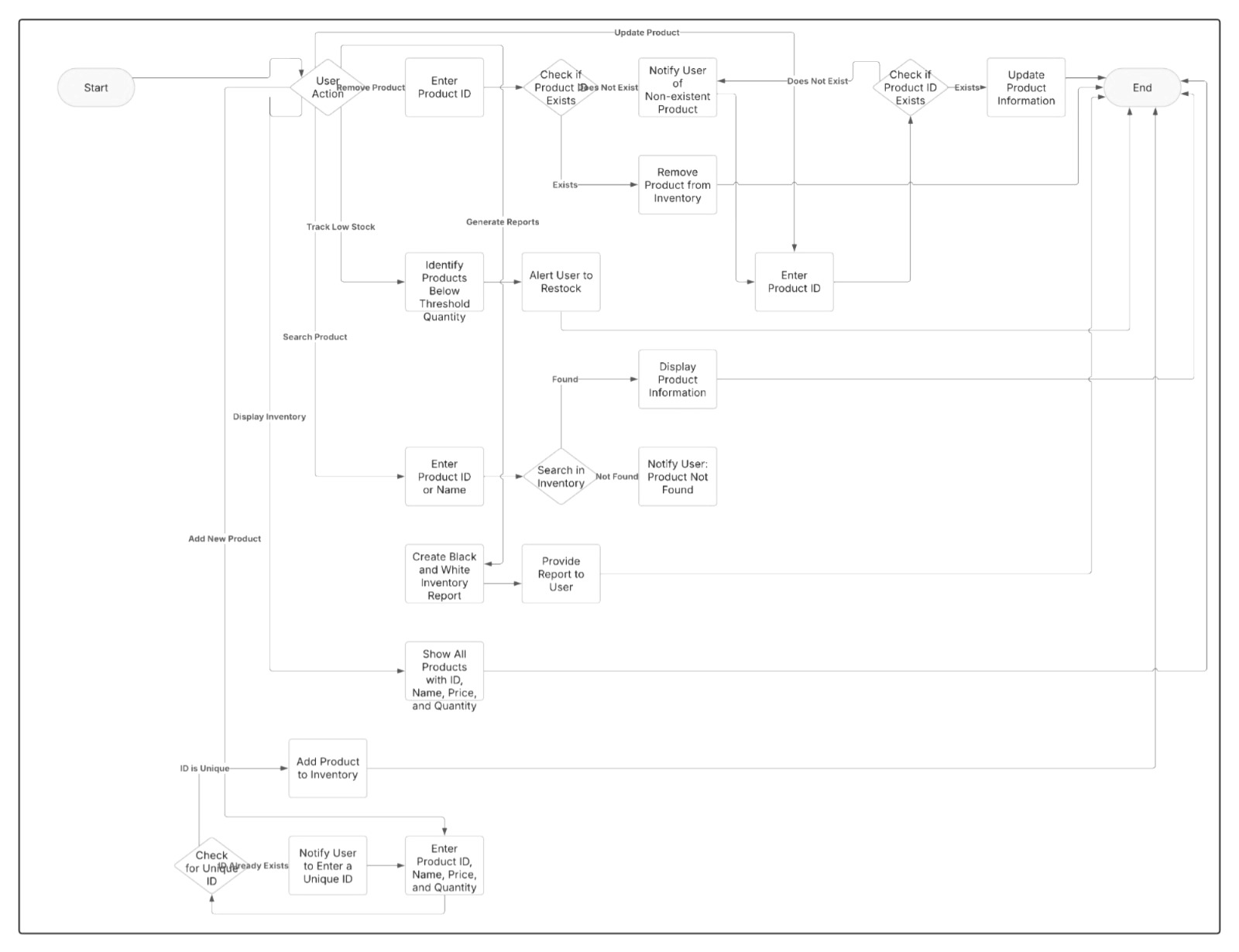
PRINT "Invalid Choice. Try again."

ENDIF

Step 4: UNTIL choice == 8

Step 5: STOP

**FLOWCHART:**



**4. Prime Number Checker**

**Program Description:** Create a program that determines whether a given number is prime or not. A prime number is only divisible by 1 and itself with no other factors.

**Key Features:**

* Accept numerical input from user
* Verify if input is valid (positive integer)
* Use efficient algorithm to check for primality
* Display result with explanation
* Option to check additional numbers

**Algorithm**

1. **Start**
2. Prompt the user to enter a positive integer.
3. Validate the input (ensure it is a positive integer).
4. If the input is **less than 2**, print "Not a prime number."
5. Check divisibility:
   * If the number is **2 or 3**, print "Prime number."
   * If the number is even or divisible by 3, print "Not a prime number."
   * Otherwise, iterate from **5 to √N** (increment by 6 each step) and check divisibility.
6. If no divisor is found, print "Prime number."
7. Ask the user if they want to check another number.
8. If **Yes**, repeat the process; if **No**, end the program.

**Pseudocode:**

Step 1: START

Step 2: DO

PRINT "Enter a positive integer:"

READ number

IF number < 2 THEN

PRINT "Not a prime number"

CONTINUE

IF number = 2 OR number = 3 THEN

PRINT "Prime number"

CONTINUE

IF number MOD 2 = 0 OR number MOD 3 = 0 THEN

PRINT "Not a prime number"

CONTINUE

isPrime ← TRUE

i ← 5

WHILE (i \* i) <= number DO

IF number MOD i = 0 OR number MOD (i + 2) = 0 THEN

isPrime ← FALSE

BREAK

ENDIF

i ← i + 6

ENDWHILE

IF isPrime THEN

PRINT "Prime number"

ELSE

PRINT "Not a prime number"

ENDIF

PRINT "Do you want to check another number? (Yes/No)"

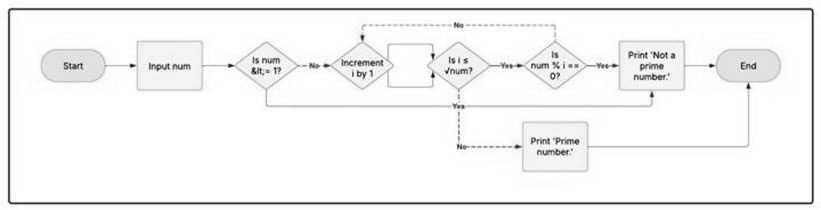
READ response

WHILE response = "Yes"

PRINT "Program Ended."

Step 3: END

**FLOWCHART:**



**5. Temperature Conversion Tool**

**Program Description:** Develop a program that converts temperatures between Celsius, Fahrenheit, and Kelvin. The user should be able to select the input and output temperature scales.

**Key Features:**

* Accept temperature value input
* Allow selection of source unit (C, F, or K)
* Allow selection of target unit (C, F, or K)
* Perform accurate conversion using correct formulas
* Display converted result with appropriate unit
* Option for multiple conversions

**Algorithm:**

Step 1: Start

Step 2: Display a menu for the user to select the source temperature unit: Celsius (C), Fahrenheit (F), or Kelvin (K).

Step 3: Accept the source temperature unit from the user.

Step 4: Prompt the user to enter the temperature value.

Step 5: Display a menu for the user to select the target temperature unit: Celsius (C), Fahrenheit (F), or Kelvin (K).

Step 6: Accept the target temperature unit from the user.

Step 7: Use conditional statements to apply the appropriate conversion formula:

* **Celsius to Fahrenheit**: F=(C×9/5)+32
* **Celsius to Kelvin**: K=C+273.15
* **Fahrenheit to Celsius**: C=(F−32)×5/9
* **Fahrenheit to Kelvin**: K=(F−32)×5/9+273.15
* **Kelvin to Celsius**: C=K−273.15
* **Kelvin to Fahrenheit**: F=(K−273.15)×9/5+32

Step 8: Display the converted temperature with the selected unit.

Step 9: Ask the user if they want to perform another conversion.

Step 10: If yes, repeat the process; otherwise, end the program.

Step 11: Stop

**Pseucode:**

Step 1: BEGIN

Step 2: DISPLAY "Temperature Conversion Tool"

Step 3: DO

DISPLAY "Select source temperature unit: (C, F, K)"

INPUT sourceUnit

DISPLAY "Enter the temperature value:"

INPUT tempValue

DISPLAY "Select target temperature unit: (C, F, K)"

INPUT targetUnit

Step 4: IF sourceUnit == "C" AND targetUnit == "F" THEN

result = (tempValue \* 9/5) + 32

ELSE IF sourceUnit == "C" AND targetUnit == "K" THEN

result = tempValue + 273.15

ELSE IF sourceUnit == "F" AND targetUnit == "C" THEN

result = (tempValue - 32) \* 5/9

ELSE IF sourceUnit == "F" AND targetUnit == "K" THEN

result = (tempValue - 32) \* 5/9 + 273.15

ELSE IF sourceUnit == "K" AND targetUnit == "C" THEN

result = tempValue - 273.15

ELSE IF sourceUnit == "K" AND targetUnit == "F" THEN

result = (tempValue - 273.15) \* 9/5 + 32

ELSE

result = tempValue // If source and target are the same

ENDIF

Step 5: DISPLAY "Converted Temperature: ", result, targetUnit

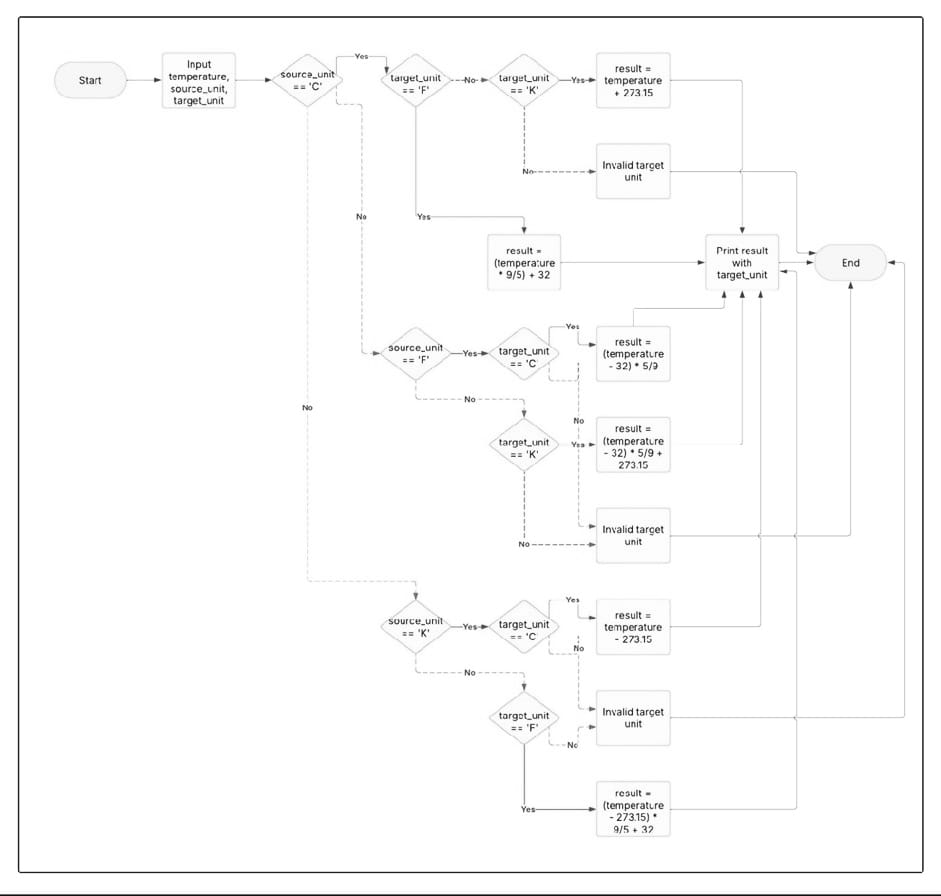
Step 6: DISPLAY "Do you want to perform another conversion? (Yes/No)"

INPUT choice

WHILE choice == "Yes"

Step 7: END

**FLOWCHART:**



**6. Library Book Management System**

**Program Description:** Design a program that manages a library's book collection, allowing librarians to add books, remove books, check out books to members, and return books. Track availability status for each book.

**Key Features:**

* Maintain database of books (title, author, ISBN, status)
* Maintain database of library members
* Process for adding new books to collection
* Process for removing obsolete books
* Book checkout procedure with due dates
* Book return procedure with potential late fees
* Search functionality by title, author, or ISBN
* Report generation for overdue books

**Algorithm:**

Step 1: Start

Step 2: Display the main menu with options:

* Add a new book
* Remove a book
* Check out a book
* Return a book
* Search for a book
* Generate overdue report
* Exit

Step 3: Accept the user’s choice.

Step 4: Based on the choice, perform the following operations:

**Adding a New Book:**

* Input book details (Title, Author, ISBN, Status = Available).
* Store book details in the database.
* Display confirmation message.

**Removing a Book:**

* Input ISBN of the book to be removed.
* Search for the book in the database.
* If found, delete the book record; otherwise, display an error message.

**Checking Out a Book:**

* Input ISBN and member ID.
* Search for the book in the database.
* If available, update status to "Checked Out" and set a due date.
* If unavailable, display an error message.

**Returning a Book:**

* Input ISBN of the book being returned.
* Search for the book in the database.
* If found, update status to "Available".
* Calculate late fee if the return date exceeds the due date.
* Display confirmation message and late fee (if applicable).

**Searching for a Book:**

* Input search criteria (Title, Author, or ISBN).
* Retrieve and display matching books.

**Generating Overdue Reports:**

* Identify books past their due date.
* Display details (Book Title, Member Name, Due Date, Late Fee).

Step 5: Ask the user if they want to perform another operation.

Step 6: If yes, repeat from Step 2; otherwise, exit.

Step 7: STOP

**Pseudocode:**

Step 1: BEGIN

Step 2: DISPLAY "Library Book Management System"

DO

DISPLAY "Select an option:"

DISPLAY "1. Add a New Book"

DISPLAY "2. Remove a Book"

DISPLAY "3. Check Out a Book"

DISPLAY "4. Return a Book"

DISPLAY "5. Search for a Book"

DISPLAY "6. Generate Overdue Report"

DISPLAY "7. Exit"

INPUT choice

Step 3: IF choice == 1 THEN

DISPLAY "Enter Book Title:"

INPUT title

DISPLAY "Enter Author Name:"

INPUT author

DISPLAY "Enter ISBN:"

INPUT isbn

status = "Available"

ADD book(title, author, isbn, status) TO libraryDatabase

DISPLAY "Book added successfully!"

ELSE IF choice == 2 THEN

DISPLAY "Enter ISBN of book to remove:"

INPUT isbn

IF bookExists(isbn) THEN

REMOVE book(isbn) FROM libraryDatabase

DISPLAY "Book removed successfully!"

ELSE

DISPLAY "Book not found!"

ELSE IF choice == 3 THEN

DISPLAY "Enter ISBN to check out:"

INPUT isbn

DISPLAY "Enter Member ID:"

INPUT memberId

IF bookAvailable(isbn) THEN

UPDATE bookStatus(isbn, "Checked Out")

dueDate = GET\_DUE\_DATE()

DISPLAY "Book checked out! Due date:", dueDate

ELSE

DISPLAY "Book is not available!"

ELSE IF choice == 4 THEN

DISPLAY "Enter ISBN of returning book:"

INPUT isbn

IF bookExists(isbn) THEN

UPDATE bookStatus(isbn, "Available")

lateFee = CALCULATE\_LATE\_FEE(isbn)

DISPLAY "Book returned successfully!"

IF lateFee > 0 THEN

DISPLAY "Late fee: $", lateFee

ELSE

DISPLAY "Book not found!"

ELSE IF choice == 5 THEN

DISPLAY "Search by: 1. Title 2. Author 3. ISBN"

INPUT searchType

DISPLAY "Enter search term:"

INPUT searchTerm

SEARCH book(searchType, searchTerm) IN libraryDatabase

DISPLAY "Matching books found: ", results

ELSE IF choice == 6 THEN

GENERATE overdueReport FROM libraryDatabase

DISPLAY overdueReport

ELSE IF choice == 7 THEN

DISPLAY "Exiting system..."

EXIT

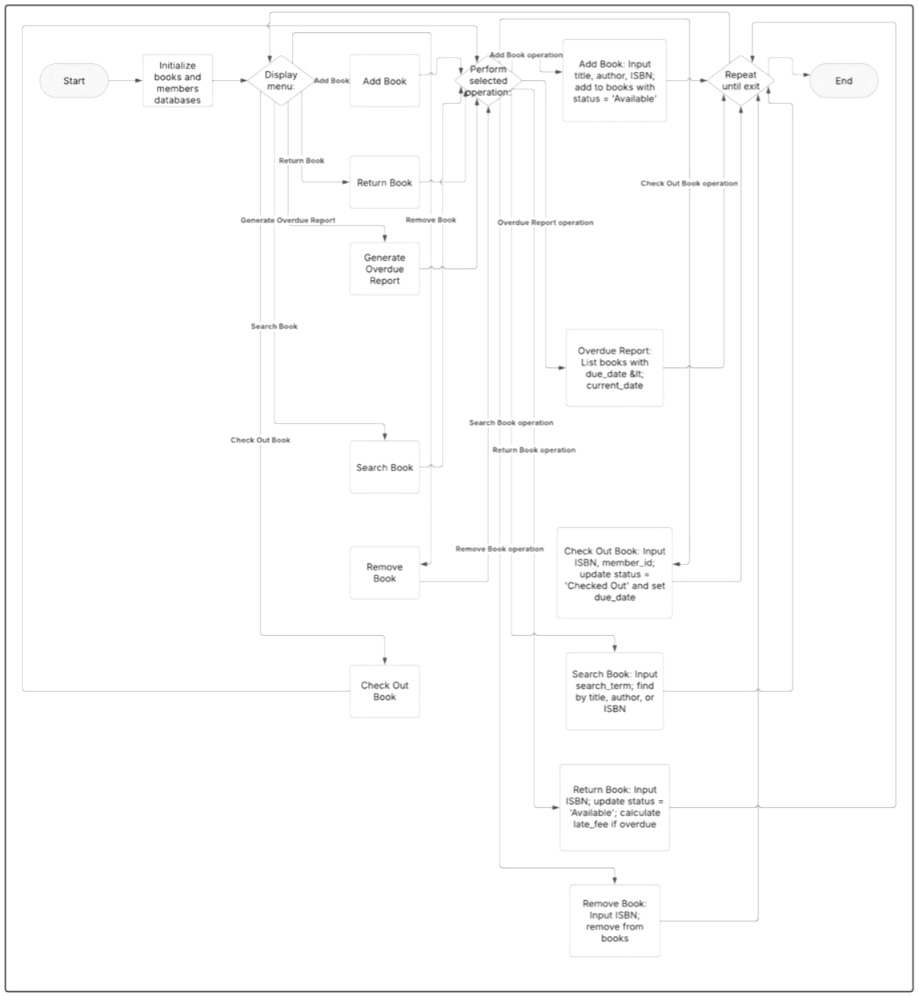
ELSE

DISPLAY "Invalid option! Please select again."

 WHILE TRUE

Step 4: END

**FLOWCHART:**



**7. Fibonacci Sequence Generator**

**Program Description:** Create a program that generates the Fibonacci sequence up to a specified number of terms. The Fibonacci sequence starts with 0 and 1, and each subsequent number is the sum of the two preceding numbers.

**Key Features:**

* Accept number of terms to generate
* Validate input is reasonable (positive integer within limits)
* Generate sequence using efficient algorithm
* Display sequence with appropriate formatting
* Option to save sequence to file

**Algorithm:**

Step 1: Start

Step 2: Input n

Step 3: Check if n is +ve

Step 4: If +ve, initialize a=0, b=1

Step 5: print a,b

Step 6: Use a loop to generate next Fibonacci numbers:

c=a+b

update a=b,b=c

print c

Step 7: Stop

**Pseudocode:**

Step 1: BEGIN

Step 2: PRINT "Enter the number of terms: "

READ n

Step 3: IF n ≤ 0 THEN

PRINT "Invalid input! Enter a positive integer."

STOP

ENDIF

Step 4: SET first = 0, second = 1

PRINT first, second

Step 5: FOR i FROM 3 TO n DO

SET next = first + second

PRINT next

SET first = second

SET second = next

ENDFOR

Step 6: PRINT "Do you want to save the sequence to a file? (yes/no)"

READ response

Step 7: IF response = "yes" THEN

OPEN file "fibonacci\_sequence.txt"

WRITE sequence to file

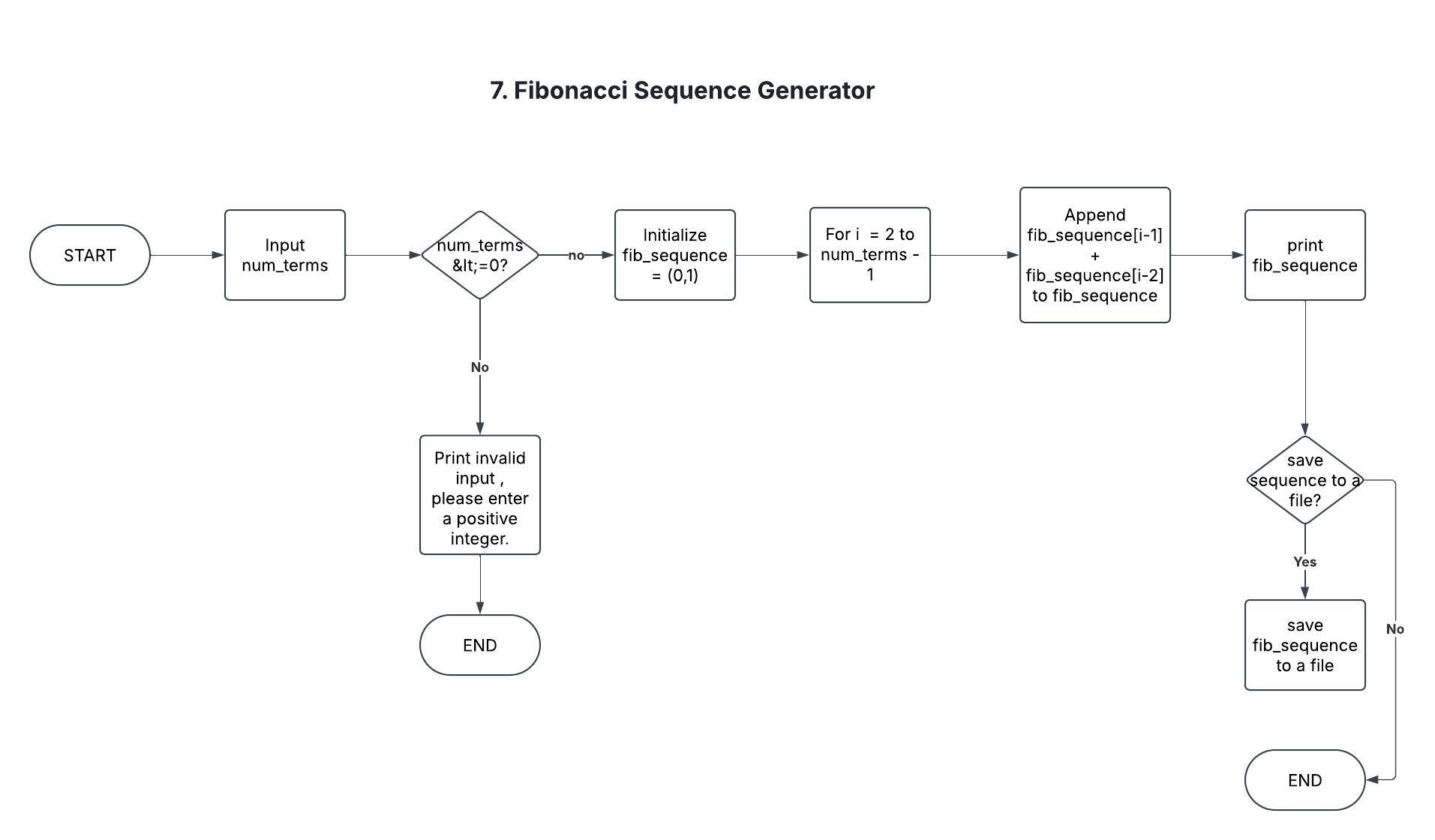
CLOSE file

PRINT "Sequence saved successfully."

  ENDIF

Step 8: END

**FLOWCHART:**

****

**8. Calendar Event Scheduler**

**Program Description: Develop a program that allows users to schedule events on a calendar. Users should be able to add events with dates, times, and descriptions, view all events, and delete events.**

**Key Features:**

* **Add events with title, date, time, and description**
* **Validate date and time inputs**
* **Store events in organized data structure**
* **Display events for a specific day, week, or month**
* **Search events by title or description**
* **Delete or modify existing events**
* **Set reminders for upcoming events**
* **Check for schedule conflicts**

**Algorithm:**

Step 1: Start

Step 2: **Initialize** an empty event list or database.

Step 3: **Display menu options**:

* Add event
* View events
* Search events
* Delete/Modify event
* Set reminders
* Exit

Step 4: **Add Event**:

* Ask the user for event title, date, time, and description.
* Validate date and time format.
* Check for schedule conflicts.
* If no conflict, store the event.

Step 5: **View Events**:

* Display events by day, week, or month.

Step 6: **Search Events**:

* Allow search by title or description.
* Display matching results.

Step 7: **Delete/Modify Events**:

* Ask for event title or date.
* Remove or modify event details.

Step 8: **Set Reminders**:

* Allow setting reminders for events.
* Notify the user when an event is near.

Step 9: STOP

**Pseudocode:**

Step 1: BEGIN

Step 2: DECLARE event\_list as an empty array

WHILE True DO

PRINT "Calendar Event Scheduler"

PRINT "1. Add Event"

PRINT "2. View Events"

PRINT "3. Search Event"

PRINT "4. Delete/Modify Event"

PRINT "5. Set Reminder"

PRINT "6. Exit"

PRINT "Choose an option: "

READ choice

Step 3: IF choice = 1 THEN

PRINT "Enter Event Title: "

READ title

PRINT "Enter Date (YYYY-MM-DD): "

READ date

PRINT "Enter Time (HH:MM): "

READ time

PRINT "Enter Description: "

READ description

Step 4: IF is\_valid\_date(date) AND is\_valid\_time(time) THEN

IF check\_conflict(date, time) = FALSE THEN

ADD (title, date, time, description) to event\_list

PRINT "Event Added Successfully."

ELSE

PRINT "Schedule Conflict! Choose a different time."

ENDIF

ELSE

PRINT "Invalid Date/Time Format."

ENDIF

ELSE IF choice = 2 THEN

PRINT "View by: 1. Day 2. Week 3. Month"

READ view\_type

DISPLAY events based on selection

ELSE IF choice = 3 THEN

PRINT "Enter search keyword: "

READ keyword

SEARCH event\_list for matching title or description

DISPLAY search results

ELSE IF choice = 4 THEN

PRINT "Enter event title or date to delete/modify: "

READ event\_key

FIND event in event\_list

IF found THEN

PRINT "1. Delete 2. Modify"

READ action

IF action = 1 THEN

REMOVE event from event\_list

PRINT "Event Deleted."

ELSE

PRINT "Enter New Details: "

UPDATE event

PRINT "Event Modified."

ENDIF

ELSE

PRINT "Event Not Found."

ENDIF

ELSE IF choice = 5 THEN

PRINT "Enter event title to set reminder: "

READ title

FIND event in event\_list

IF found THEN

PRINT "Enter reminder time before event (minutes): "

READ reminder\_time

SET reminder

PRINT "Reminder Set."

ELSE

PRINT "Event Not Found."

ENDIF

ELSE IF choice = 6 THEN

PRINT "Exiting Program."

BREAK

ELSE

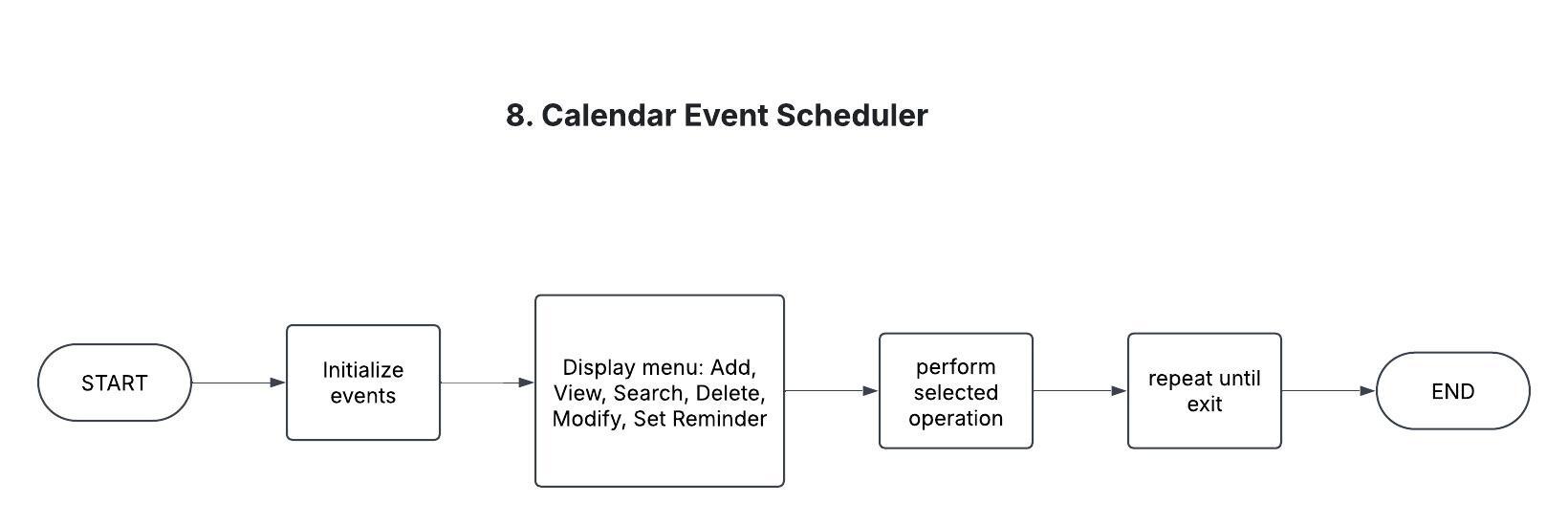
PRINT "Invalid Option. Try Again."

ENDIF

    ENDWHILE

Step 5: END

**FLOWCHART:**

****