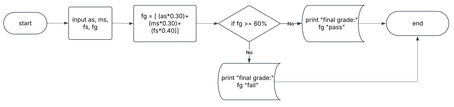
1. STUDENT GRADE CALCULATOR

Algorithm
start
input assignment score as, midterm score ms, final score fs
final grade fg =[(as*0.30)+(ms*0.30)+(fs*0.40)]
if fg>= 60% then print "pass"
else "false"
end

PSEUDOCODE
START
INPUT assignment_score, midterm_score,final_score, final_grade
PRINT "Enter assignment score : ", assignment_score
PRINT "Enter midterm score : ", midterm_score
final_score = assignment_score + midterm_score
PRINT " Enter final score : ", final_score
final_grade = [(assignment_score*0.30)+(midterm_score*0.30)+(final_score*0.40)]
IF final_grade>=60% then
status = "pass"
else
status = "fail"
END IF
PRINT "Final Grade is : ", final_grade
PRINT " Status : ", status
END

FLOWCHART

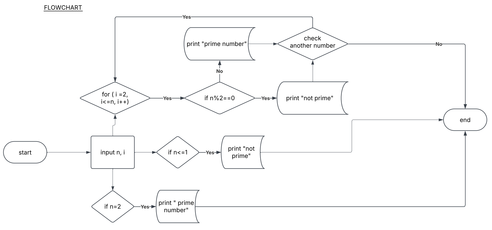


4. PRIME NUMBER CHECKER

Algorithm
Start
Input: Read the number n.
Check Special Cases: If n<1 return Not Prime.
If n=2 return Prime (2 is the only even prime number).
Loop from if (n % 2== 0)
n is divisible by any number in this range, return Not Prime.
If no divisors found, return Prime.
End

PSEUDOCODE
START
INPUT n, i
PRINT " Enter n value: "
IF n<=1 then
PRINT " Not prime"
ELSE IF n = 2 then
PRINT "Prime number"
ELSE isPrime ← TRUE
FOR (i=2, i<=n, i++)
IF n%2==0 THEN
isPrime ← FALSE
BREAK
END IF
END FOR
IF isPrime THEN
PRINT n, "is a Prime Number"
ELSE PRINT n, "is Not a Prime Number"
END IF PRINT "Do you want to check another number? (yes/no)"
READ response
UNTIL response = "no"
PRINT "Program Ended" END

FLOWCHART



2. ATM BANKING SYSTEM

ALGORITHM
Start
input PIN, check balance, deposit, withdraw, exit
1. Authenticate user with PIN
2. Display menu options:
3.
    - Check balance
    - Deposit money
    - Withdraw money (only if sufficient balance)
    - Exit
4. Process user selection:
    - If Check Balance, display current balance
    - If Deposit, add amount to balance and print receipt
    - If Withdraw, check if balance is sufficient:
    -
        - If yes, deduct amount and print receipt
        - If no, show "Insufficient Funds"
    - If Exit, terminate the program
5. Loop menu until user exits

end

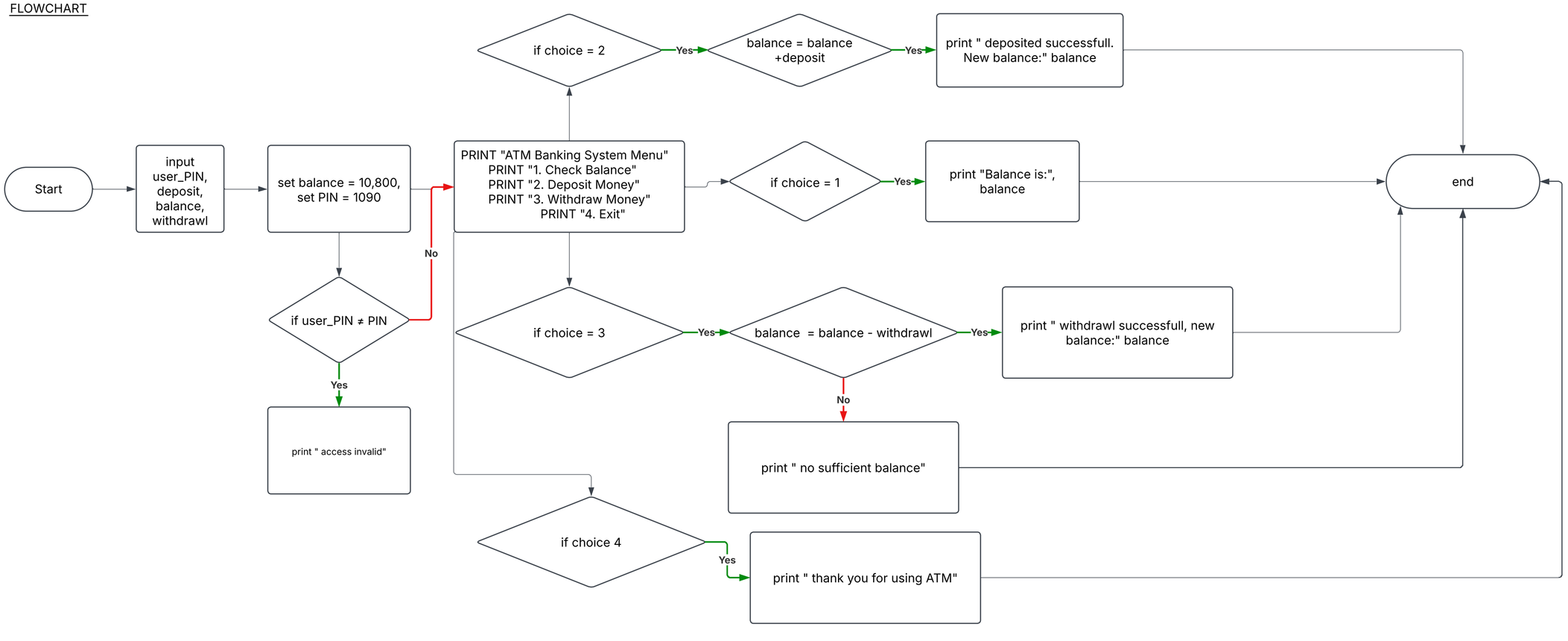
PSEUDOCODE

```
START
   SET balance = 10,800
   SET PIN = 1090
INPUT user_PIN, deposit, withdrawl
   PRINT "Enter your PIN:"
   READ user_PIN
   IF user_PIN ≠ PIN THEN
      PRINT "Invalid PIN. Access Denied."
         END IF
   REPEAT
      PRINT "ATM Banking System Menu"
      PRINT "1. Check Balance"
      PRINT "2. Deposit Money"
      PRINT "3. Withdraw Money"
      PRINT "4. Exit"

      PRINT "Enter your choice:"
      READ choice

      IF choice = 1 THEN
         PRINT "Your current balance is: ", balance
      ELSE IF choice = 2 THEN
         PRINT "Enter deposit amount:"
         READ deposit
         balance = balance + deposit
         PRINT "Deposit successful. New balance: ", balance
      ELSE IF choice = 3 THEN
         PRINT "Enter withdrawal amount:"
         READ withdraw
         IF withdraw ≤ balance THEN
            balance = balance - withdraw
            PRINT "Withdrawal successful. New balance: ", balance
         ELSE
            PRINT "Insufficient Funds!"
         END IF
      ELSE IF choice = 4 THEN
         PRINT "Thank you for using the ATM. Goodbye!"
         EXIT PROGRAM
      ELSE
         PRINT "Invalid choice. Please select again."
      END IF
     UNTIL choice = 4
END
```

FLOWCHART



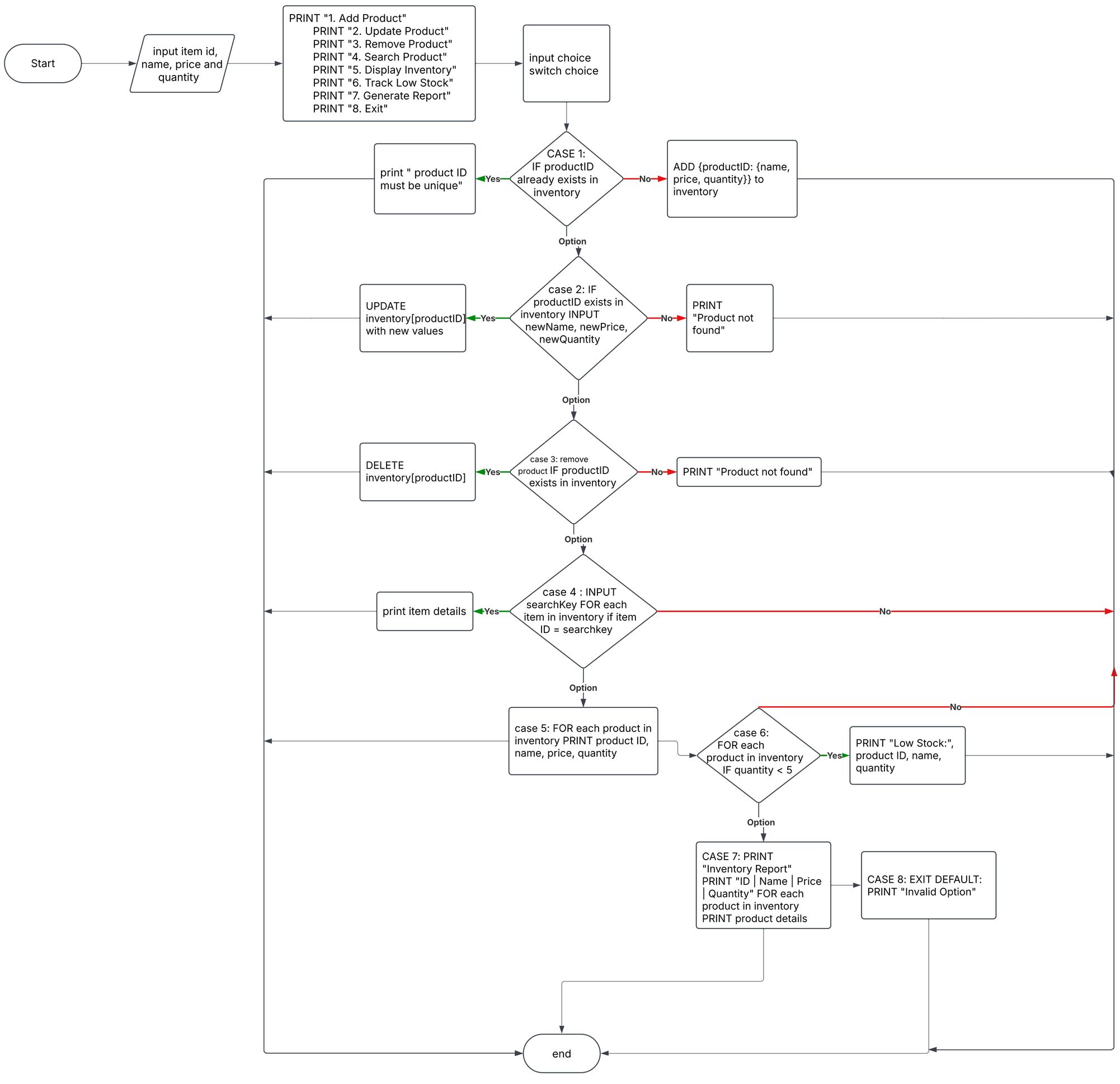
4. INVENTORY MANAGEMENT SYSTEM

ALGORITHM
Start
create dictionary of store inventory items
input of item id, name, price and quantity
1. Display Menu Options
   - Add New Product
   - Update Product Information
   - Remove Product
   - Search Product (by ID or Name)
   - Display Inventory
   - Track Low Stock Items
   - Generate Inventory Report
   - Exit
2. Perform Actions Based on User Input
   - Add New Product:
     - Prompt user for product details (ID, Name, Price, Quantity).
     - Store details in the inventory list.
     - Ensure unique ID constraint.
   - Update Product Information:
     - Search product by ID.
     - If found, allow updating Name, Price, or Quantity.
   - Remove Product:
     - Search product by ID.
     - If found, remove it from the inventory.
   - Search Product:
     - Allow searching by ID or Name.
     - Display product details if found.
   - Display Inventory:
     - Print all stored products in a table format.
   - Track Low Stock Items:
     - Identify and list items with quantity below a predefined threshold (e.g.,
   - Generate Inventory Report:
     - Display a structured report of all products, their quantities, and values.
   - Exit Program

PSEUDOCODE

```
START
    DECLARE inventory as Dictionary
        WHILE TRUE
        PRINT "1. Add Product"
        PRINT "2. Update Product"
        PRINT "3. Remove Product"
        PRINT "4. Search Product"
        PRINT "5. Display Inventory"
        PRINT "6. Track Low Stock"
        PRINT "7. Generate Report"
        PRINT "8. Exit"
       INPUT choice
        SWITCH choice
            CASE 1:
                INPUT productID, name, price, quantity
                IF productID already exists in inventory
                    PRINT "Error: Product ID must be unique"
                ELSE
                    ADD {productID: {name, price, quantity}} to inventory
            CASE 2:
                INPUT productID
                IF productID exists in inventory
                    INPUT newName, newPrice, newQuantity
                    UPDATE inventory[productID] with new values
                ELSE
                    PRINT "Product not found"
            CASE 3:
                INPUT productID
                IF productID exists in inventory
                    DELETE inventory[productID]
                ELSE
                    PRINT "Product not found"
            CASE 4:
                INPUT searchKey
                FOR each item in inventory
                    IF item ID or name matches searchKey
                        PRINT item details
            CASE 5:
                FOR each product in inventory
                    PRINT product ID, name, price, quantity
            CASE 6:
                FOR each product in inventory
                    IF quantity < 5
                        PRINT "Low Stock:", product ID, name, quantity
            CASE 7:
                PRINT "Inventory Report"
                PRINT "ID | Name | Price | Quantity"
                FOR each product in inventory
                    PRINT product details
            CASE 8:
                EXIT
            DEFAULT:
                PRINT "Invalid Option"
END
```

FLOWCHART



5. TEMPERATURE CONVERSION TOOL

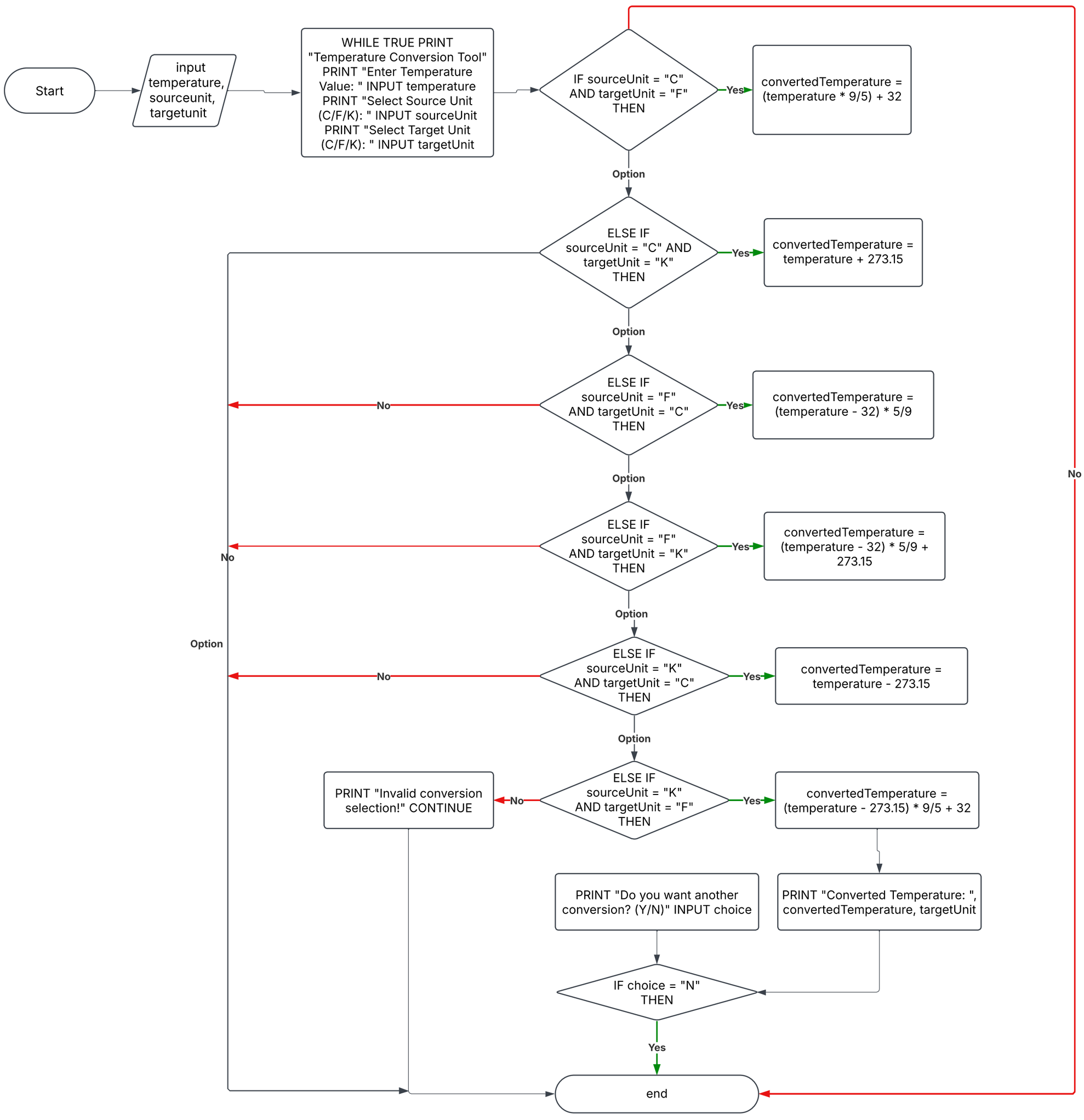
Algorithm
start
1. Display options for the user:
   - Input a temperature value.
   - Select the source unit (C, F, or K).
   - Select the target unit (C, F, or K).
2. Perform temperature conversion using the correct formulas:
   - Celsius (C) to Fahrenheit (F): F=(C×95)+32F=(C×59)+32
   - Celsius (C) to Kelvin (K): K=C+273.15K=C+273.15
   - Fahrenheit (F) to Celsius (C): C=(F−32)×59C=(F−32)×95
   - Fahrenheit (F) to Kelvin (K): K=(F−32)×59+273.15K=(F−32)×95+273.15
   - Kelvin (K) to Celsius (C): C=K−273.15C=K−273.15
   - Kelvin (K) to Fahrenheit (F): F=(K−273.15)×95+32F=(K−273.15)×59+32
3. Display the converted temperature with the appropriate unit.
4. Ask the user if they want another conversion.
   - If yes, repeat the process.
   - If no, exit the program.

end

PSEUDOCODE

```
START
   WHILE TRUE
      PRINT "Temperature Conversion Tool"
      PRINT "Enter Temperature Value: "
      INPUT temperature
      PRINT "Select Source Unit (C/F/K): "
      INPUT sourceUnit
      PRINT "Select Target Unit (C/F/K): "
      INPUT targetUnit
      IF sourceUnit = "C" AND targetUnit = "F" THEN
         convertedTemperature = (temperature * 9/5) + 32
      ELSE IF sourceUnit = "C" AND targetUnit = "K" THEN
         convertedTemperature = temperature + 273.15
      ELSE IF sourceUnit = "F" AND targetUnit = "C" THEN
         convertedTemperature = (temperature - 32) * 5/9
      ELSE IF sourceUnit = "F" AND targetUnit = "K" THEN
         convertedTemperature = (temperature - 32) * 5/9 + 273.15
      ELSE IF sourceUnit = "K" AND targetUnit = "C" THEN
         convertedTemperature = temperature - 273.15
      ELSE IF sourceUnit = "K" AND targetUnit = "F" THEN
         convertedTemperature = (temperature - 273.15) * 9/5 + 32
      ELSE
         PRINT "Invalid conversion selection!"
         CONTINUE
      PRINT "Converted Temperature: ", convertedTemperature,
targetUnit
      PRINT "Do you want another conversion? (Y/N)"
      INPUT choice
      IF choice = "N" THEN
         EXIT
   END WHILE
END
```

FLOWCHART



6. LIBRARY BOOK MANAGEMENT SYSTEM

Algorithm

start

1. Initialize book and member databases:
   - Each book has a title, author, ISBN, and availability status.
   - Each member has a name, ID, and borrowed books list.
2. Display menu options:
   - Add a book.
   - Remove a book.
   - Search for a book.
   - Check out a book.
   - Return a book.
   - Generate overdue report.
   - Exit.
3. User selects an option:
   - Add Book: Enter title, author, ISBN, and set status to "Available".
   - Remove Book: Search by ISBN and remove if it exists.
   - Search Book: Find by title, author, or ISBN.
   - Check Out Book:
     - Search book by ISBN.
     - If available, assign it to a member and update the due date.
     - Mark book status as "Checked Out".
   - Return Book:
     - Search for the book.
     - If overdue, calculate late fees.
     - Mark book status as "Available".
   - Overdue Report:
     - Display all books with due dates past the current date.
4. Ask user if they want to continue:
5. If yes, go back to menu.
   - If no, exit.

end

PSEUDOCODE

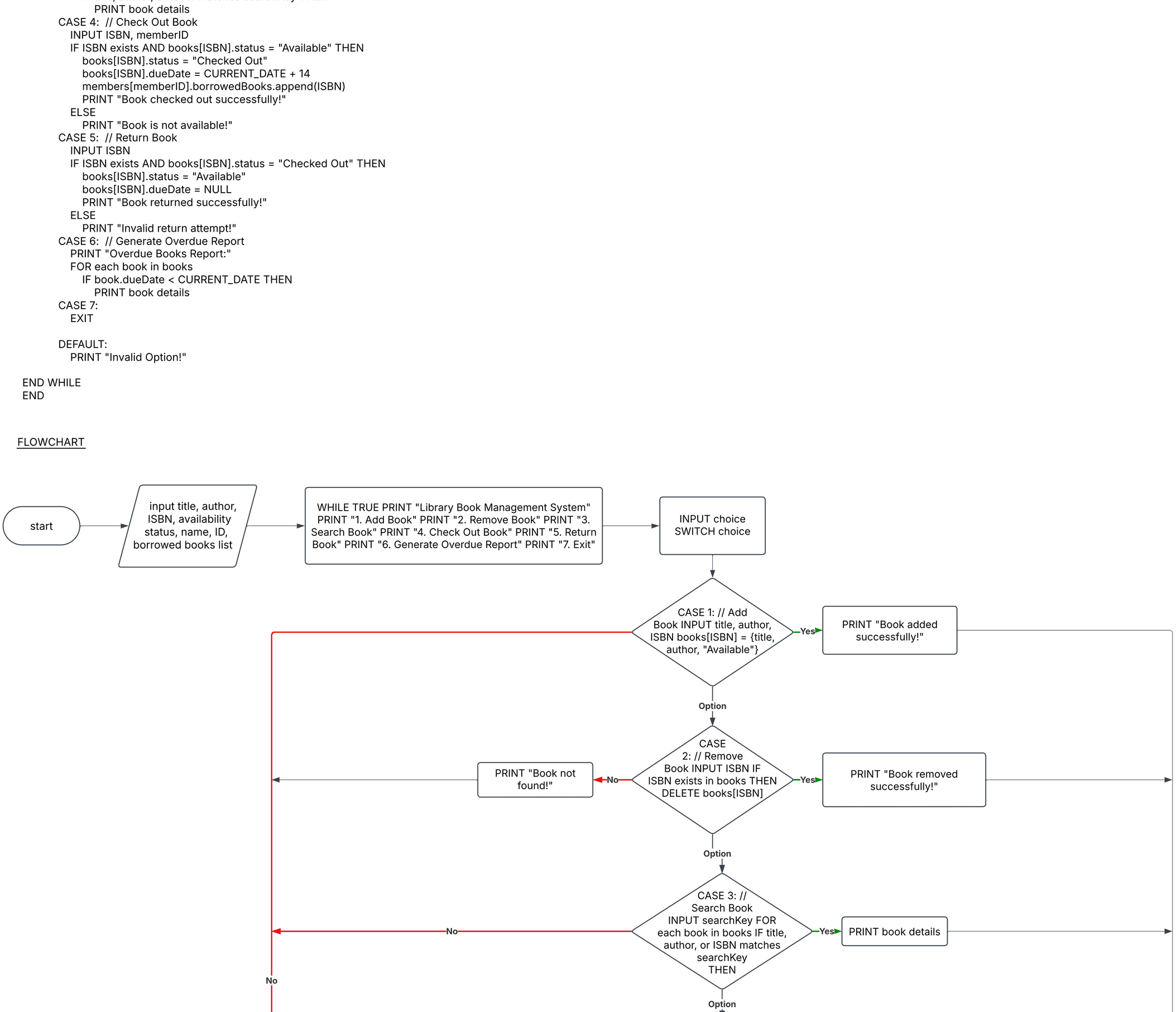
```
START
    DECLARE books as Dictionary
    DECLARE members as Dictionary
    WHILE TRUE
        PRINT "Library Book Management System"
        PRINT "1. Add Book"
        PRINT "2. Remove Book"
        PRINT "3. Search Book"
        PRINT "4. Check Out Book"
        PRINT "5. Return Book"
        PRINT "6. Generate Overdue Report"
        PRINT "7. Exit"

        INPUT choice
        SWITCH choice
            CASE 1:  // Add Book
                INPUT title, author, ISBN
                books[ISBN] = {title, author, "Available"}
                PRINT "Book added successfully!"
            CASE 2:  // Remove Book
                INPUT ISBN
                IF ISBN exists in books THEN
                    DELETE books[ISBN]
                    PRINT "Book removed successfully!"
                ELSE
                    PRINT "Book not found!"
             CASE 3:  // Search Book
                INPUT searchKey
                FOR each book in books
                    IF title, author, or ISBN matches searchKey THEN
                        PRINT book details
            CASE 4:  // Check Out Book
                INPUT ISBN, memberID
                IF ISBN exists AND books[ISBN].status = "Available" THEN
                    books[ISBN].status = "Checked Out"
                    books[ISBN].dueDate = CURRENT_DATE + 14
                    members[memberID].borrowedBooks.append(ISBN)
                    PRINT "Book checked out successfully!"
                ELSE
                    PRINT "Book is not available!"
            CASE 5:  // Return Book
                INPUT ISBN
                IF ISBN exists AND books[ISBN].status = "Checked Out" THEN
                    books[ISBN].status = "Available"
                    books[ISBN].dueDate = NULL
                    PRINT "Book returned successfully!"
                ELSE
                    PRINT "Invalid return attempt!"
            CASE 6:  // Generate Overdue Report
                PRINT "Overdue Books Report:"
                FOR each book in books
                    IF book.dueDate < CURRENT_DATE THEN
                        PRINT book details
            CASE 7:
                EXIT

            DEFAULT:
                PRINT "Invalid Option!"

    END WHILE
    END
```

FLOWCHART



7. FIBONACCI SEQUENCE GENERATOR

Algorithm

start

1. Input the number of terms (N).
2. Validate the input:
   - Check if N is a positive integer.
   - Ensure N is within a reasonable limit (e.g., max 50 terms).
3. Generate the Fibonacci sequence:
   - If N = 1, output [0].
   - If N = 2, output [0, 1].
   - For N > 2, compute each term using: $F(n)=F(n-1)+F(n-2)$
   - Display the sequence in a formatted manner.
4. Ask the user if they want to save the sequence to a file.
   - If yes, save the sequence to a text file.
5. Ask the user if they want to generate another sequence.
   - If yes, repeat from Step 2.
   - If no, exit.

end

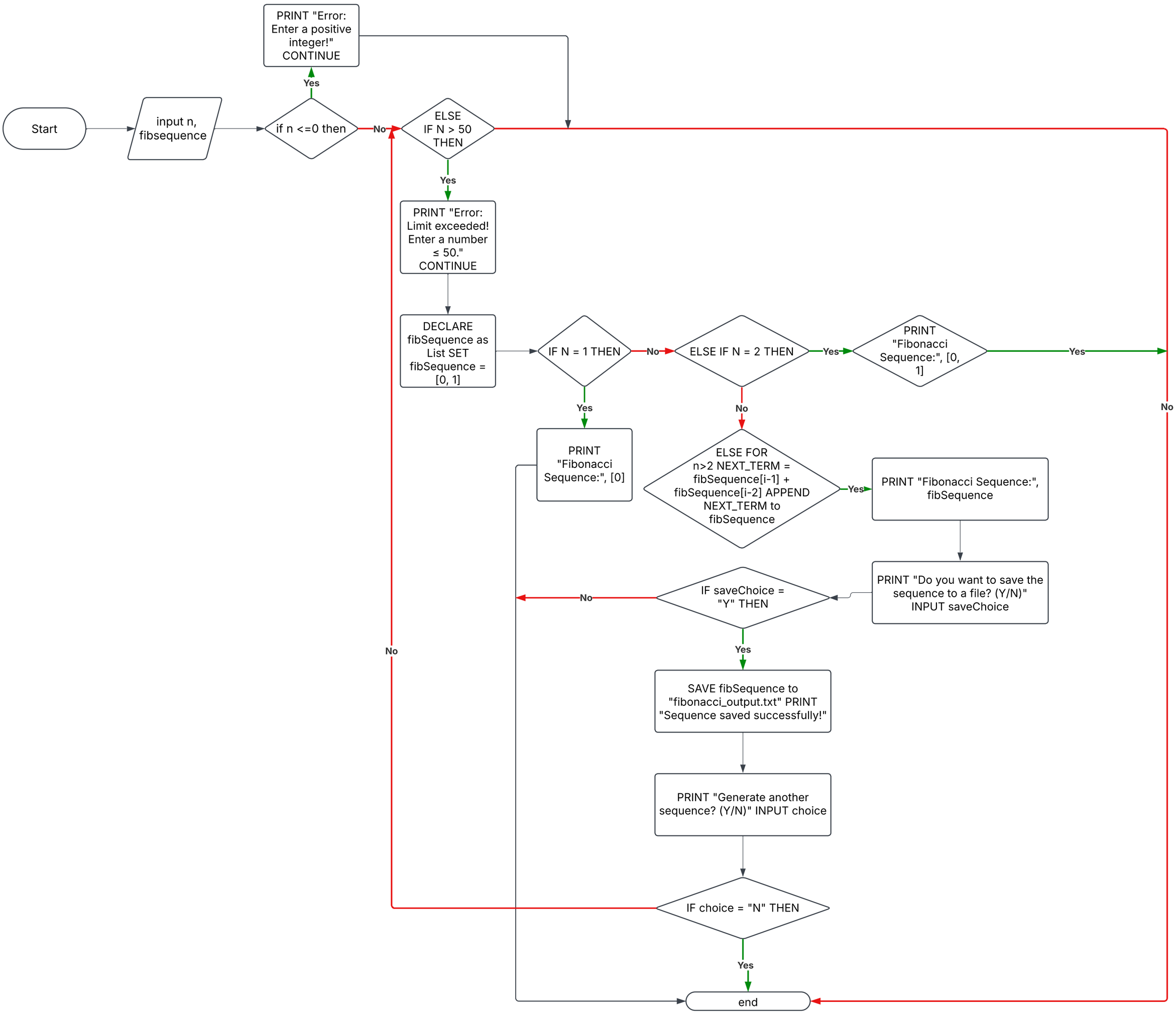
PSEUDOCODE

```
START
   WHILE TRUE
      PRINT "Fibonacci Sequence Generator"
      INPUT N
      IF N <= 0 THEN
         PRINT "Error: Enter a positive integer!"
         CONTINUE
      ELSE IF N > 50 THEN
         PRINT "Error: Limit exceeded! Enter a number ≤ 50."
         CONTINUE
      DECLARE fibSequence as List
      SET fibSequence = [0, 1]
      IF N = 1 THEN
         PRINT "Fibonacci Sequence:", [0]
      ELSE IF N = 2 THEN
         PRINT "Fibonacci Sequence:", [0, 1]
      ELSE
         FOR i FROM 2 TO N-1
            NEXT_TERM = fibSequence[i-1] + fibSequence[i-2]
            APPEND NEXT_TERM to fibSequence

         PRINT "Fibonacci Sequence:", fibSequence

      PRINT "Do you want to save the sequence to a file? (Y/N)"
      INPUT saveChoice
      IF saveChoice = "Y" THEN
         SAVE fibSequence to "fibonacci_output.txt"
         PRINT "Sequence saved successfully!"
      PRINT "Generate another sequence? (Y/N)"
      INPUT choice
      IF choice = "N" THEN
         EXIT
END WHILE
END
```

FLOWCHART



8. CALENDAR EVENT SCHEDULER

Algorithm
start
1. Create an event database (Dictionary or List) to
   store event details.
2. Display the main menu with options.

User selects an option

1. Add an event:
   - Prompt user for title, date, time, and description.
   - Validate date and time format.
   - Check for schedule conflicts.
   - If no conflict, store the event.
2. View events:
   - Allow users to view events for a specific day,
     week, or month.
   - Retrieve and display events from the database.
3. Search for an event:
   - Allow users to search by title or description.
   - Display matching events.
4. Modify an event:
   - Allow users to edit an event's title, date, time, or
     description.
   - Check for conflicts before saving changes.
5. Delete an event:
   - Prompt for the event title or date.
   - If found, remove it from the database.
6. Set reminders:
   - Check upcoming events.
   - Notify the user X minutes/hours before.

end

PSEUDOCODE

```
START
   DECLARE events as Dictionary
   WHILE TRUE
      PRINT "1. Add Event"
      PRINT "2. View Events"
      PRINT "3. Search Events"
      PRINT "4. Modify Event"
      PRINT "5. Delete Event"
      PRINT "6. Set Reminders"
      PRINT "7. Exit"

      INPUT choice
      SWITCH choice
         CASE 1: addEvent()
INPUT title, date, time, description
      IF date OR time is invalid THEN
         PRINT "Invalid date or time format!"
         RETURN
      IF checkConflict(date, time) THEN
         PRINT "Schedule Conflict! Choose another time."
      ELSE
         ADD {title, date, time, description} to events
         PRINT "Event added successfully!"

         CASE 2: viewEvents()
INPUT choice (day/week/month)
      DISPLAY events matching the choice

         CASE 3: searchEvent()
 INPUT searchKey
      FOR each event in events
         IF event.title OR event.description matches searchKey THEN
            PRINT event details

         CASE 4: modifyEvent()
INPUT eventTitle
      IF event exists THEN
         INPUT newTitle, newDate, newTime, newDescription
         UPDATE event details
         PRINT "Event updated successfully!"
      ELSE
         PRINT "Event not found!"

         CASE 5: deleteEvent()
 INPUT eventTitle
      IF event exists THEN
         DELETE event
         PRINT "Event deleted successfully!"
      ELSE
         PRINT "Event not found!"

         CASE 6: setReminder()
 FOR each upcoming event
         IF event.time is approaching THEN
            PRINT "Reminder: Event at", event.time

         CASE 7: checkConflict(date, time)
 FOR each event in events
         IF event.date == date AND event.time == time THEN
            RETURN TRUE
      RETURN FALSE
   case 8: EXIT

      DEFAULT: PRINT "Invalid Option!"
   END
```

FLOWCHART