**1. Student Grade Calculator**

START

PRINT "Enter the score for assignments (out of 100): "

READ assignments\_score

PRINT "Enter the score for midterm exam (out of 100): "

READ midterm\_score

PRINT "Enter the score for final exam (out of 100): "

READ final\_exam\_score

assignments\_weight = 0.30

midterm\_weight = 0.30

final\_exam\_weight = 0.40

final\_grade = (assignments\_score \* assignments\_weight) +

(midterm\_score \* midterm\_weight) +

(final\_exam\_score \* final\_exam\_weight)

IF final\_grade >= 60 THEN

status = "Pass"

ELSE

status = "Fail"

END IF

PRINT "Final Grade: ", final\_grade

PRINT "Status: ", status

END

**Algorithm**

start

Get the student's score for assignments, midterm exam, and final exam.

Assign a weight of 30% for assignments, 30% for the midterm exam, and 40% for the final exam.

Calculate the final grade using the formula:

inal grade=(assignments\_score×0.30)+(midterm\_score\*0.30)+(final\_exam\_score×0.40)

If the final grade is greater than or equal to 60, print "Pass".

If the final grade is less than 60, print "Fail".

Display the final grade and the pass/fail status.

end.

**2. ATM Banking System**

START

SET balance = 1000

SET pin = 1234

PRINT "Enter your PIN: "

READ entered\_pin

IF entered\_pin != pin THEN

PRINT "Incorrect PIN. Access denied."

EXIT

END IF

REPEAT

PRINT "ATM 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 the amount to deposit: "

READ deposit\_amount

IF deposit\_amount <= 0 THEN

PRINT "Invalid amount. Deposit must be positive."

ELSE

balance = balance + deposit\_amount

PRINT "Deposit successful. Your new balance is: ", balance

END IF

ELSE IF choice == 3 THEN

PRINT "Enter the amount to withdraw: "

READ withdraw\_amount

IF withdraw\_amount <= 0 THEN

PRINT "Invalid amount. Withdrawal must be positive."

ELSE IF withdraw\_amount > balance THEN

PRINT "Insufficient balance. Withdrawal denied."

ELSE

balance = balance - withdraw\_amount

PRINT "Withdrawal successful. Your new balance is: ", balance

PRINT "Transaction Receipt: Withdrawal of ", withdraw\_amount, " completed."

END IF

ELSE IF choice == 4 THEN

PRINT "Thank you for using the ATM. Goodbye!"

EXIT

ELSE

PRINT "Invalid choice. Please enter a valid option."

END IF

UNTIL choice == 4 // Repeat the process until the user chooses to exit

END

**Algorithm**

Ask the user to input their PIN.

If the entered PIN matches the stored PIN, grant access to the ATM. Otherwise, deny access.

Show the available options: Check balance, Deposit money, Withdraw money, Exit.

Check Balance: Display the current balance.

Deposit Money: Prompt the user to enter the deposit amount and update the balance.

Withdraw Money: Prompt the user to enter the withdrawal amount, validate that the amount doesn't exceed the available balance, and update the balance if the withdrawal is allowed.

Exit: End the session and thank the user for using the ATM.

Continue displaying the menu until the user selects "Exit."

End

**3. Inventory Management System**

START

INVENTORY = {}

REPEAT

PRINT "Inventory Management System Menu:"

PRINT "1. Add New Product"

PRINT "2. Update Product Information"

PRINT "3. Remove Product"

PRINT "4. Search Product"

PRINT "5. Display Inventory"

PRINT "6. Track Low Stock Items"

PRINT "7. Generate Report"

PRINT "8. Exit"

READ user\_choice

IF user\_choice == 1 THEN

PRINT "Enter Product ID:"

READ product\_id

PRINT "Enter Product Name:"

READ product\_name

PRINT "Enter Product Price:"

READ product\_price

PRINT "Enter Product Quantity:"

READ product\_quantity

IF product\_id exists in INVENTORY THEN

PRINT "Product already exists. Choose to update instead."

ELSE

ADD product\_id, product\_name, product\_price, product\_quantity TO INVENTORY

PRINT "Product added successfully!"

END IF

ELSE IF user\_choice == 2 THEN

PRINT "Enter Product ID to update:"

READ product\_id

IF product\_id exists in INVENTORY THEN

PRINT "Enter new name (leave blank to keep current):"

READ new\_name

IF new\_name is not empty THEN

UPDATE product\_name to new\_name

END IF

PRINT "Enter new price (leave blank to keep current):"

READ new\_price

IF new\_price is not empty THEN

UPDATE product\_price to new\_price

END IF

PRINT "Enter new quantity (leave blank to keep current):"

READ new\_quantity

IF new\_quantity is not empty THEN

UPDATE product\_quantity to new\_quantity

END IF

PRINT "Product updated successfully!"

ELSE

PRINT "Product ID not found."

END IF

ELSE IF user\_choice == 3 THEN

PRINT "Enter Product ID to remove:"

READ product\_id

IF product\_id exists in INVENTORY THEN

REMOVE product\_id FROM INVENTORY

PRINT "Product removed successfully!"

ELSE

PRINT "Product ID not found."

END IF

ELSE IF user\_choice == 4 THEN

PRINT "Enter Product ID or Name to search:"

READ search\_query

FOUND = FALSE

FOR each product in INVENTORY:

IF product\_id or product\_name matches search\_query THEN

PRINT product details

FOUND = TRUE

END IF

END FOR

IF FOUND == FALSE THEN

PRINT "Product not found."

END IF

ELSE IF user\_choice == 5 THEN

PRINT "Current Inventory:"

FOR each product in INVENTORY:

PRINT product details (ID, Name, Price, Quantity)

ELSE IF user\_choice == 6 THEN

PRINT "Low stock items (Quantity <= 10):"

FOR each product in INVENTORY:

IF product\_quantity <= 10 THEN

PRINT product details (ID, Name, Quantity)

END IF

END FOR

ELSE IF user\_choice == 7 THEN

total\_value = 0

total\_products = 0

FOR each product in INVENTORY:

total\_value = total\_value + (product\_price \* product\_quantity)

total\_products = total\_products + 1

END FOR

PRINT "Total number of products: total\_products"

PRINT "Total inventory value: total\_value"

ELSE IF user\_choice == 8 THEN

PRINT "Exiting Inventory Management System."

BREAK

ELSE

PRINT "Invalid choice. Please try again."

UNTIL user\_choice == 8

END

**Algorithm**

Start.

Initialize an empty list or dictionary called INVENTORY to store product details.

Display Menu with the following options:

Add New Product

Update Product Information

Remove Product

Search Product

Display Inventory

Track Low Stock Items

Generate Report

Exit

Repeat the following until the user selects "Exit" (Option 8):

Read the user’s choice.

If the user chooses to add a new product:

Input Product ID, Name, Price, and Quantity.

If Product ID already exists in INVENTORY:

Display "Product already exists".

Else:

Add product to INVENTORY.

Display "Product added successfully".

Else If the user chooses to update product information:

Input Product ID to update.

If the product exists:

Input new name, price, or quantity.

Update the product in INVENTORY.

Display "Product updated successfully".

Else:

Display "Product not found".

Else If the user chooses to remove a product:

Input Product ID to remove.

If the product exists:

Remove the product from INVENTORY.

Display "Product removed successfully".

Else:

Display "Product not found".

Else If the user chooses to search for a product:

Input Product ID or Name.

If the product exists:

Display product details.

Else:

Display "Product not found".

Else If the user chooses to display inventory:

Display all products in INVENTORY.

Else If the user chooses to track low stock items (quantity ≤ 10):

Display all products with quantity less than or equal to 10.

Else If the user chooses to generate a report:

Calculate and display the total number of products and the total value of the inventory.

Exit the system.

End.

**4. Prime Number Checker**

START

PRINT "Enter a positive integer to check if it is prime: "

READ number

IF number <= 1 THEN

PRINT "Please enter a positive integer greater than 1."

ELSE

IS\_PRIME = TRUE

FOR i = 2 to square root of number:

IF number mod i == 0 THEN

IS\_PRIME = FALSE

BREAK

END IF

END FOR

IF IS\_PRIME == TRUE THEN

PRINT number, "is a prime number."

ELSE

PRINT number, "is not a prime number."

END IF

END IF

PRINT "Do you want to check another number? (yes/no): "

READ response

IF response == "yes" THEN

REPEAT the process

ELSE

PRINT "Exiting the program."

EXIT

END IF

END

**Algorithm**

Start.

Prompt the user to input a number.

Check if the number is less than or equal to 1:

If yes, display "Please enter a positive integer greater than 1."

If no, proceed to step 4.

Check if the number is prime:

Set a flag IS\_PRIME to True.

Loop i from 2 to the square root of the number:

If the number is divisible by i, set IS\_PRIME to False and exit the loop.

If IS\_PRIME is True, display "The number is prime."

If IS\_PRIME is False, display "The number is not prime."

If the user says "yes", go back to step 2.

If the user says "no", exit the program.

End.

**5. Temperature Conversion Tool**

START

PRINT "Enter the temperature value:"

READ temperature\_value

PRINT "Select the source unit (C for Celsius, F for Fahrenheit, K for Kelvin):"

READ source\_unit

PRINT "Select the target unit (C for Celsius, F for Fahrenheit, K for Kelvin):"

READ target\_unit

IF source\_unit == "C" THEN

IF target\_unit == "F" THEN

target\_temperature = (temperature\_value \* 9/5) + 32

PRINT "Converted temperature: ", target\_temperature, " Fahrenheit"

ELSE IF target\_unit == "K" THEN

target\_temperature = temperature\_value + 273.15

PRINT "Converted temperature: ", target\_temperature, " Kelvin"

END IF

ELSE IF source\_unit == "F" THEN

IF target\_unit == "C" THEN

target\_temperature = (temperature\_value - 32) \* 5/9

PRINT "Converted temperature: ", target\_temperature, " Celsius"

ELSE IF target\_unit == "K" THEN

target\_temperature = ((temperature\_value - 32) \* 5/9) + 273.15

PRINT "Converted temperature: ", target\_temperature, " Kelvin"

END IF

ELSE IF source\_unit == "K" THEN

IF target\_unit == "C" THEN

target\_temperature = temperature\_value - 273.15

PRINT "Converted temperature: ", target\_temperature, " Celsius"

ELSE IF target\_unit == "F" THEN

target\_temperature = ((temperature\_value - 273.15) \* 9/5) + 32

PRINT "Converted temperature: ", target\_temperature, " Fahrenheit"

END IF

END IF

PRINT "Do you want to perform another conversion? (yes/no):"

READ response

IF response == "yes" THEN

REPEAT the process

ELSE

PRINT "Exiting the program."

EXIT

END IF

END

**Algorithm**

Start.

Prompt the user to enter the temperature value.

Prompt the user to select the source unit (C for Celsius, F for Fahrenheit, K for Kelvin).

Prompt the user to select the target unit (C for Celsius, F for Fahrenheit, K for Kelvin).

Perform conversion based on the selected source and target units:

f the source unit is Celsius (C):

Convert to Fahrenheit: F = (C \* 9/5) + 32

Convert to Kelvin: K = C + 273.15

If the source unit is Fahrenheit (F):

Convert to Celsius: C = (F - 32) \* 5/9

Convert to Kelvin: K = ((F - 32) \* 5/9) + 273.15

If the source unit is Kelvin (K):

Convert to Celsius: C = K - 273.15

Convert to Fahrenheit: F = ((K - 273.15) \* 9/5) + 32

Display the converted result with the appropriate unit.

Ask the user if they want to perform another conversion:

If yes, go back to step 2.

If no, exit the program.

End.

**6. Library Book Management System**

START

BOOKS\_DATABASE = {} // A dictionary to store book details (ISBN, Title, Author, Status)

MEMBERS\_DATABASE = {} // A dictionary to store member details (MemberID, Name)

REPEAT

PRINT "Library Book Management System"

PRINT "1. Add New Book"

PRINT "2. Remove Book"

PRINT "3. Checkout Book"

PRINT "4. Return Book"

PRINT "5. Search Book"

PRINT "6. Generate Overdue Report"

PRINT "7. Exit"

READ user\_choice

IF user\_choice == 1 THEN

PRINT "Enter Book Title:"

READ title

PRINT "Enter Book Author:"

READ author

PRINT "Enter Book ISBN:"

READ isbn

IF isbn not in BOOKS\_DATABASE THEN

BOOKS\_DATABASE[isbn] = {"title": title, "author": author, "status": "available", "due\_date": None}

PRINT "Book added successfully."

ELSE

PRINT "Book with this ISBN already exists."

ELSE IF user\_choice == 2 THEN

PRINT "Enter Book ISBN to remove:"

READ isbn

IF isbn in BOOKS\_DATABASE THEN

REMOVE isbn from BOOKS\_DATABASE

PRINT "Book removed successfully."

ELSE

PRINT "Book not found."

ELSE IF user\_choice == 3 THEN

PRINT "Enter Member ID:"

READ member\_id

PRINT "Enter Book ISBN to checkout:"

READ isbn

IF isbn in BOOKS\_DATABASE AND BOOKS\_DATABASE[isbn]["status"] == "available" THEN

PRINT "Enter due date (YYYY-MM-DD):"

READ due\_date

BOOKS\_DATABASE[isbn]["status"] = "checked\_out"

BOOKS\_DATABASE[isbn]["due\_date"] = due\_date

PRINT "Book checked out successfully."

ELSE

PRINT "Book is not available for checkout or does not exist."

ELSE IF user\_choice == 4 THEN

PRINT "Enter Book ISBN to return:"

READ isbn

IF isbn in BOOKS\_DATABASE AND BOOKS\_DATABASE[isbn]["status"] == "checked\_out" THEN

current\_date = GET\_CURRENT\_DATE()

due\_date = BOOKS\_DATABASE[isbn]["due\_date"]

IF current\_date > due\_date THEN

late\_days = CALCULATE\_LATE\_DAYS(current\_date, due\_date)

late\_fee = late\_days \* 1 // Example: $1 per day late

PRINT "Book is overdue. Late fee: $", late\_fee

ELSE

PRINT "Book returned on time."

END IF

BOOKS\_DATABASE[isbn]["status"] = "available"

BOOKS\_DATABASE[isbn]["due\_date"] = None

ELSE

PRINT "Book was not checked out or does not exist."

ELSE IF user\_choice == 5 THEN

PRINT "Search by (1) Title, (2) Author, (3) ISBN:"

READ search\_choice

IF search\_choice == 1 THEN

PRINT "Enter Title to search:"

READ title

FOR each book in BOOKS\_DATABASE:

IF BOOKS\_DATABASE[book]["title"] == title THEN

PRINT book details

END IF

ELSE IF search\_choice == 2 THEN

PRINT "Enter Author to search:"

READ author

FOR each book in BOOKS\_DATABASE:

IF BOOKS\_DATABASE[book]["author"] == author THEN

PRINT book details

END IF

ELSE IF search\_choice == 3 THEN

PRINT "Enter ISBN to search:"

READ isbn

IF isbn in BOOKS\_DATABASE THEN

PRINT book details

ELSE

PRINT "Book not found."

ELSE IF user\_choice == 6 THEN

PRINT "Overdue Books Report:"

current\_date = GET\_CURRENT\_DATE()

FOR each book in BOOKS\_DATABASE:

IF BOOKS\_DATABASE[book]["status"] == "checked\_out" AND current\_date > BOOKS\_DATABASE[book]["due\_date"] THEN

PRINT book details

PRINT "Overdue by: ", CALCULATE\_LATE\_DAYS(current\_date, BOOKS\_DATABASE[book]["due\_date"]), " days"

END IF

ELSE IF user\_choice == 7 THEN

PRINT "Exiting Library Management System."

BREAK

ELSE

PRINT "Invalid choice, try again."

UNTIL user\_choice == 7

END

**Algorithm**

Start.

Initialize the books and members databases.

Display the main menu with options to:

Add a new book.

Remove a book.

Checkout a book.

Return a book.

Search for a book.

Generate overdue report.

Exit.If adding a book:

Input book details (title, author, ISBN).

Add the book to the database if it doesn't already exist.

If removing a book:

Input ISBN.

Remove the book from the database if it exists.

If checking out a book:

Input member ID and ISBN.

If the book is available, mark it as checked out and set a due date.

If returning a book:

Input ISBN.

If the book is checked out, calculate any late fees if applicable and return the book.

If searching for a book:

Search by title, author, or ISBN and display matching books.

If generating an overdue report:

Check each checked-out book and display overdue books.

Repeat the menu until the user chooses to exit.

End.

**7. Fibonacci Sequence Generator**

START

PRINT "Enter the number of terms for Fibonacci sequence:"

READ num\_terms

IF num\_terms <= 0 THEN

PRINT "Please enter a positive integer greater than 0."

ELSE IF num\_terms > LIMIT THEN

PRINT "Please enter a number less than or equal to the maximum limit."

ELSE

first\_term = 0

second\_term = 1

PRINT first\_term

PRINT second\_term

FOR i = 3 to num\_terms DO

next\_term = first\_term + second\_term

PRINT next\_term

first\_term = second\_term

second\_term = next\_term

END FOR

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

READ response

IF response == "yes" THEN

OPEN a file to save the sequence

FOR i = 1 to num\_terms DO

WRITE the Fibonacci number to the file

END FOR

CLOSE the file

PRINT "Sequence saved to file."

ELSE

PRINT "Sequence not saved."

END IF

END IF

END

**Algorithm**

Start.

Prompt the user to input the number of terms in the Fibonacci sequence.

Validate the input:

If the input is less than or equal to 0, display "Please enter a positive integer greater than 0."

If the input exceeds a reasonable limit (e.g., 1000), display "Please enter a number less than or equal to the maximum limit."

first\_term = 0

second\_term = 1

Display the first two terms (0 and 1).

Loop from 3 to the number of terms specified.

Calculate the next term as the sum of the previous two terms.

Update the previous two terms to the last two numbers in the sequence.

If yes, open a file and write the Fibonacci sequence to the file.

If no, just print the sequence to the screen.

End.

**8. Calendar Event Scheduler**

START

EVENTS\_DATABASE = {} // {date: [(time, title, description, reminder)]}

REPEAT

PRINT "Calendar Event Scheduler"

PRINT "1. Add Event"

PRINT "2. View Events"

PRINT "3. Delete Event"

PRINT "4. Search Events"

PRINT "5. Exit"

READ user\_choice

IF user\_choice == 1 THEN

PRINT "Enter event title:"

READ title

PRINT "Enter event date (YYYY-MM-DD):"

READ event\_date

PRINT "Enter event time (HH:MM):"

READ event\_time

PRINT "Enter event description:"

READ description

IF NOT is\_valid\_date(event\_date) THEN

PRINT "Invalid date format. Please use YYYY-MM-DD."

CONTINUE

END IF

IF NOT is\_valid\_time(event\_time) THEN

PRINT "Invalid time format. Please use HH:MM."

CONTINUE

END IF

IF event\_date not in EVENTS\_DATABASE THEN

EVENTS\_DATABASE[event\_date] = []

END IF

EVENTS\_DATABASE[event\_date].append((event\_time, title, description, None)) // No reminder set yet

PRINT "Event added successfully."

ELSE IF user\_choice == 2 THEN

PRINT "Enter date to view events (YYYY-MM-DD):"

READ view\_date

IF view\_date in EVENTS\_DATABASE THEN

PRINT "Events for", view\_date, ":"

FOR each event in EVENTS\_DATABASE[view\_date]:

PRINT event

ELSE

PRINT "No events found for this date."

ELSE IF user\_choice == 3 THEN

PRINT "Enter date of event to delete (YYYY-MM-DD):"

READ event\_date

PRINT "Enter time of event to delete (HH:MM):"

READ event\_time

IF event\_date in EVENTS\_DATABASE THEN

FOR each event in EVENTS\_DATABASE[event\_date]:

IF event[0] == event\_time THEN

EVENTS\_DATABASE[event\_date].remove(event)

PRINT "Event deleted successfully."

BREAK

END FOR

ELSE

PRINT "Event not found."

ELSE IF user\_choice == 4 THEN

PRINT "Enter search keyword (title or description):"

READ search\_keyword

PRINT "Search results:"

FOR each date in EVENTS\_DATABASE:

FOR each event in EVENTS\_DATABASE[date]:

IF search\_keyword in event[1] OR search\_keyword in event[2] THEN

PRINT date, event

END IF

END FOR

ELSE IF user\_choice == 5 THEN

PRINT "Exiting Calendar Event Scheduler."

BREAK

ELSE

PRINT "Invalid choice, try again."

UNTIL user\_choice == 5

END

FUNCTION is\_valid\_date(date):

RETURN date matches the format

FUNCTION is\_valid\_time(time):

RETURN time matches the format

**Algorithm**

Start.

Initialize an empty data structure (e.g., a dictionary EVENTS\_DATABASE) to store events, where the key is the date (YYYY-MM-DD) and the value is a list of events. Each event is represented as a tuple of (time, title, description, reminder).

Add an event.

View events for a specific date.

Delete an event.

Search events by title or description.

Exit.

If adding an event:

Prompt the user to input the title, date, time, and description.

Validate the date and time input.

Store the event in the database under the given date.

If viewing events:

Prompt the user to input a date.

Display all events for that date if any exist.

If deleting an event:

Prompt the user to input the date and time of the event to delete.

Find and remove the event from the database if it exists.

If searching events:

Prompt the user to input a search keyword (for title or description).

Search the database and display matching events.

If exiting:

Exit the program.

Repeat the menu until the user chooses to exit.

End.