Stock management system

Overviews

- ✓Introduction
- ✓Problem statements
- ✓Pseudo code
- ✓Flowchart diagram
- ✓C++ codes and outputs
- ✓Conclusion
- ✓References

Introduction

- Stock management system is a valuable and essential tool for any type of business. As business progress to expand and develop their outcomes, the demand for stock management system becomes rising important.
- Our project states an advanced solution by generating a console based application in C++, using object- concepts

Problem statements

- Manual stock management system creates:-
- shortages, wastages, overstocking, understocking of stocks
- Time consumption
- Wastage of costs
- Inefficient resource utilization
- Unstable economy
- Gradual economic development

Pseudo code

Stock items = empty list

Function add_stock_item(inventory):

Name = input(“Enter the name of the item: “)

Quantity = input(“Enter the quantity of the item: “)

Price = input(“Enter the price of the item: “)

New item = create_stock_item(name, quantity, price)

Append new_item to inventory

Print “Item added successfully!”

Function display_ inventory(inventory):
    Print “Current Inventory:”
    For item in inventory:
        Print “Name: “ + item.name + “, Quantity: “ + item.quantity + “, Price: “ + item.price

Function calculate_item_prices(inventory):
    Print “Item Prices:”
    For item in inventory:
        Item_price = item.quantity * item.price
        Print “Name: “ + item.name + “, Item Price: “ + item_price

Function calculate_total_price(inventory):
    Total_price = 0

For item in inventory:
Item_price = item.quantity * item.price
Total_price = total_price + item_price
Return total_price
Function update_quantity(inventory):
Name = input(“Enter the name of the item to update: “)
New_quantity = input(“Enter the new quantity: “)
Found = false
For item in inventory:
If item.name is equal to name:
Item.quantity = new_quantity
Found = true
Break

If found is true:
Print “Quantity updated successfully!”
Else:
Print “Item not found!”
Function remove_stock_item(inventory):
Name = input(“Enter the name of the item to remove: “)
Found = false
For item in inventory:
If item.name is equal to name:
Remove item from inventory
Found = true
Break
If found is true:
Print “Item removed successfully!”
Else:
Print “Item not found!”

Function save_to_file(inventory):
    File = open(“inventory.txt”, “w”)
    For item in inventory:
        Write item.name to file
        Write item.quantity to file
        Write item.price to file
Close file
    Print “Inventory saved to file successfully!”
Function read_from_file(inventory):
    File = open(“inventory.txt”, “r”)
    For line in file:
        Split line into name, quantity, price
        Create new_item with name, quantity, price
        Append new_item to inventory
    Close file
    Print “Inventory loaded from file successfully

Function main():

Inventory = empty list

Choice = -1

While choice is not equal to 9:

Print “Stock Management System”

Print “----------”

Print “1. Add Stock Item”

Print “2. Display Inventory”

Print “3. Calculate Item Prices”

Print “4. Calculate Total Price”

Print “5. Update Quantity”

Print “6. Remove Stock Item”

Print “7. Save Inventory to File”

Print “8. Load Inventory from File”

Print “9. Exit”

Choice = input(“Enter your choice: “)

Switch choice:

Case 1:

Add_stock_item(inventory)

Break

Case 2:

Display_inventory(inventory)

Break

Case 3:

Calculate_item_prices(inventory)

Break

Case 4:

Total_price = calculate_total_price(inventory)

Print “Total Price of all Items: “ + total_price

Break

Case 5:

Update_quantity(inventory)

Break

Case 6:

Remove_stock_item(inventory)

Break

Case 7:

Save_to_file(inventory)

Break

Case 8:

Read_from_file(inventory)

Break

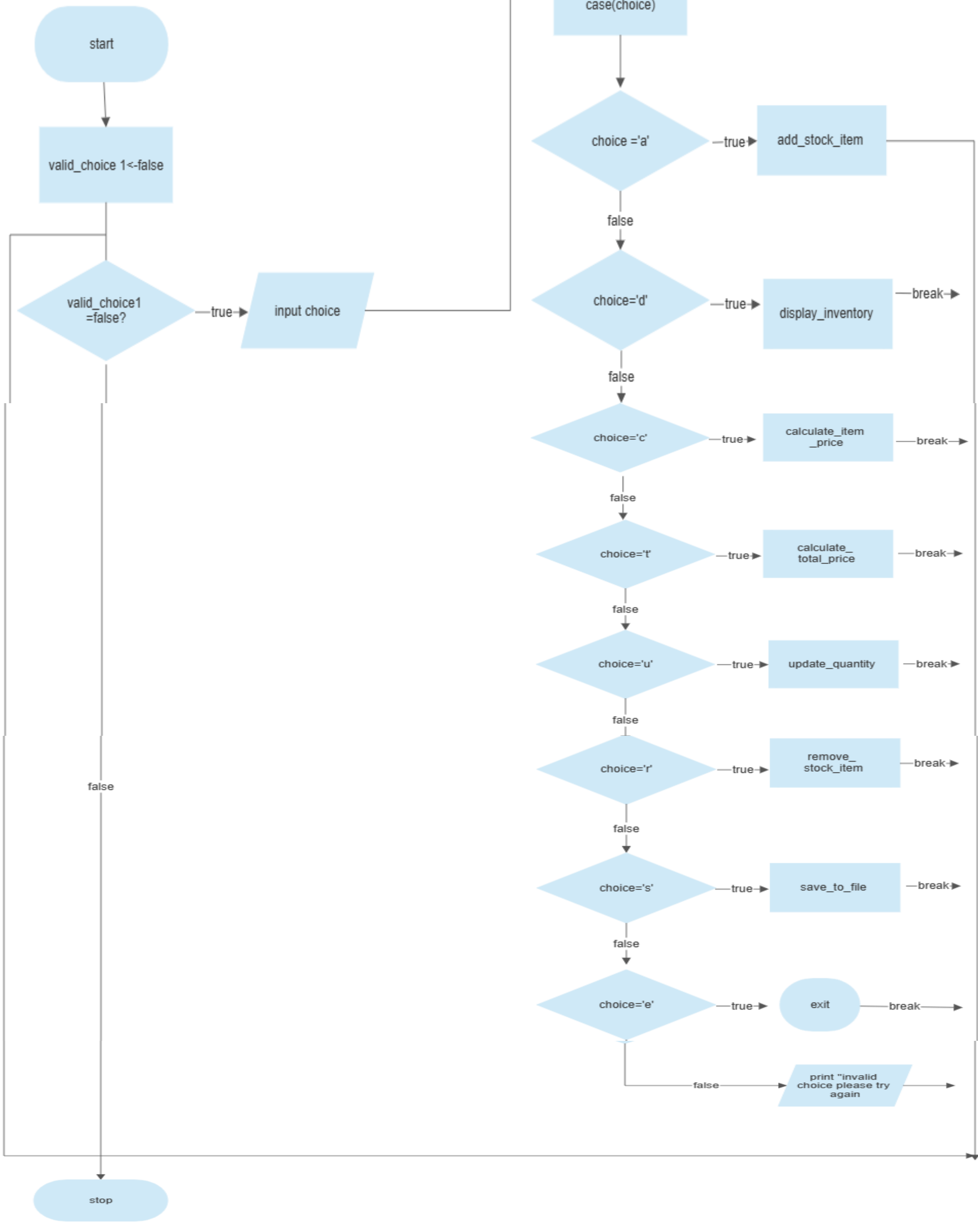
Case 9:

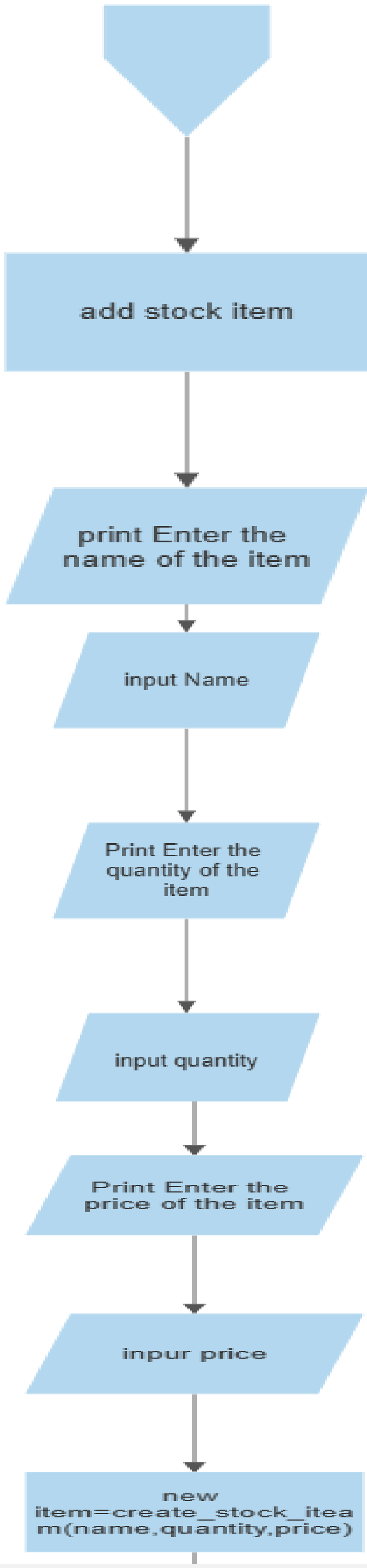
Print “Exiting…”

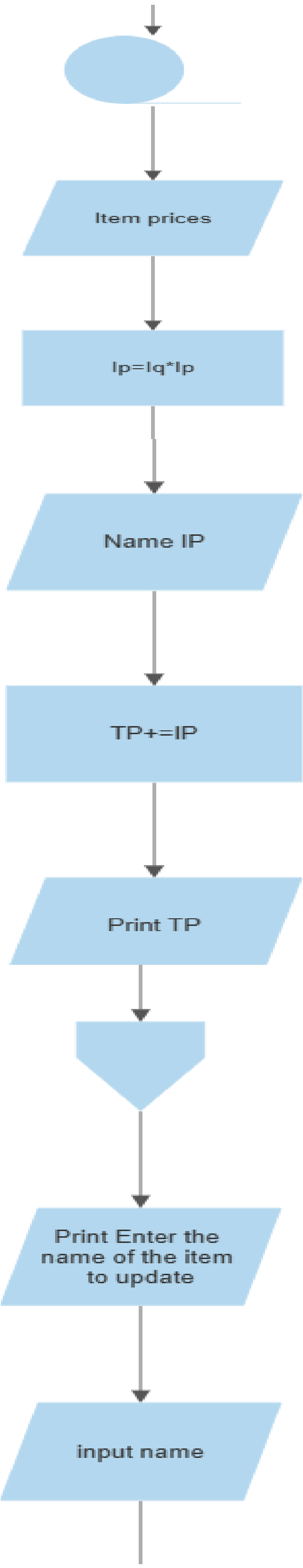
Break

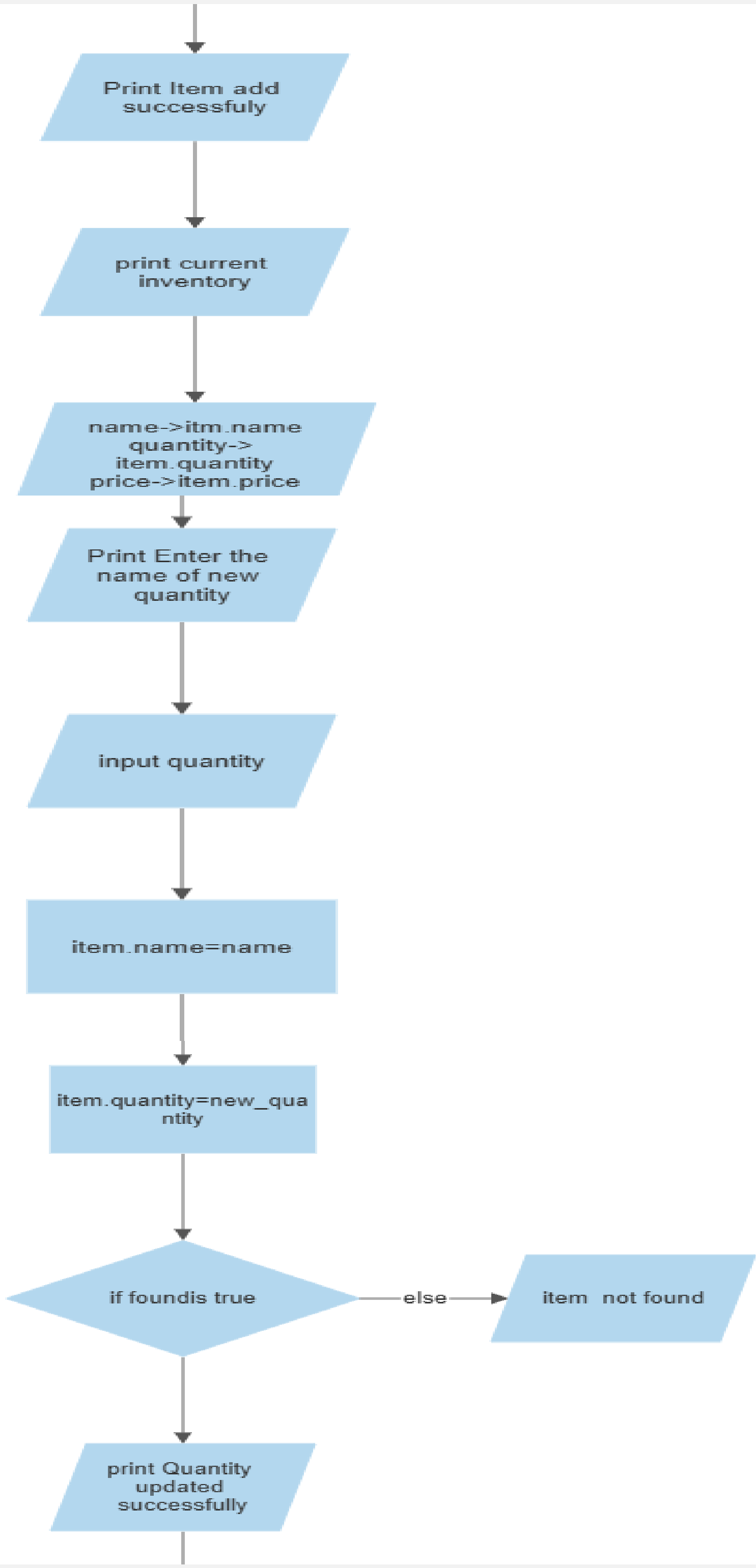
Default:

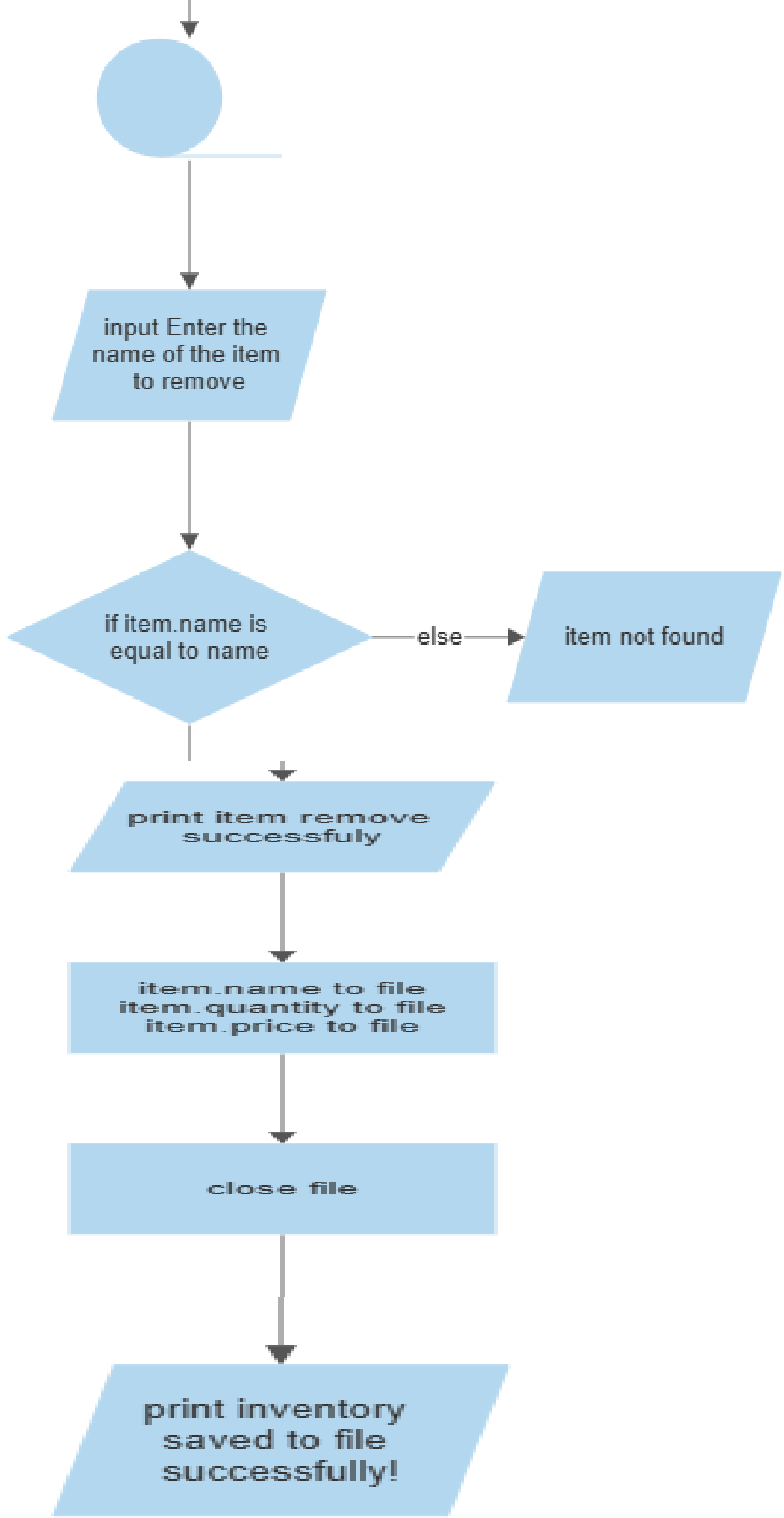
Print “Invalid choice. Please try again.”


start
valid_choice 1<-false
valid_choice1 =false?
true
input choice
choice<-upper case(choice)
choice ='a'
true
add_stock_item
false
choice='d'
true
display_inventory
break
false
choice='c'
true
calculate_item _price
break
false
choice='t'
true
calculate_ total_price
break
false
choice='u'
true
update_quantity
break
false
choice='r'
true
remove_ stock_item
break
false
choice='s'
true
save_to_file
break
false
choice='e'
true
exit
break
false
print "invalid choice please try again
false
stop


add stock item
print Enter the name of the item
input Name
Print Enter the quantity of the item
input quantity
Print Enter the price of the item
inpur price
new item=create_stock_itea m(name,quantity,price)


Item prices
lp=lq*lp
Name IP
TP+=IP
Print TP
Print Enter the name of the item to update
input name


Print Item add successfuly
print current inventory
name->itm.name
quantity->
item.quantity
price->item.price
Print Enter the name of new quantity
input quantity
item.name=name
item.quantity=new_quantity
if foundis true
else
item not found
print Quantity updated successfully


input Enter the name of the item to remove
if item.name is equal to name
else
item not found
print item remove successfuly
item.name to file
item.quantity to file
item.price to file
close file
print inventory saved to file successfully!

Conclusion

- The development of stock management system for Addis Ababa city using C++ and systems based on object oriented concepts is an important mechanism to increase productivity and effectiveness in the city's stock management system. The project's objective is to provide convenient and practical solution for businesses and organizations in Addis Ababa to efficiently manage their stocks, enhance security of inventory control.

References

- Deitel, P. J. (2017). How to Program C++ (2017 edition). Boston, MA: Pearson.
- Meyer, B. (1997). Object-Oriented Software Construction (2nd Ed.). Upper Saddle River, NJ: Prentice Hall.
- Stroustrup, B. (2013). The C++ Programming Language (4th Ed.). Boston, MA: Addison-Wesley.

- Alamy Images, “I,” Alamy Stock Photos, accessed 6 9, 2023 https://www.alamy.com/