FUNCTION customerSearch ( string \*array, string array[], integer len)

FOR(integer i=0i<len i++)

IF arr[i] == key

return i

return -1

END FUNCTION

FUNCTION checkQty(integer \*arr,integer key, integer len)

FOR(integer j=0 j<lenj++)

IF arr[j] == key

return j

return -1

END FUNCTION

FUNCTION checkItemNumber(integer \*arr,integer \*key, integer len)

FOR (integer j=0j<lenj++)

IF(arr[j]==key[j])

return j

return -1

END FUNCTION

FUNCTION integer updateQty(integer qtyOld, integer qtyNew)

integer finalQty=qtyOld-qtyNew

IF (finalQty<0||finalQty>qtyOld)

OUTPUT There is not enough of this product

ELSE

return finalQty

END FUNCTION

FUNCTION double updateCreditLimit(double oldCredit, double newCredit)

double finalCredit=oldCredit-newCredit

IF (finalCredit <0 OR finalCredit > oldCredit)

OUTPUT There is not enough money in the account endl

ELSE

return finalCredit

END FUNCTION

FUNCTION string generateOrderNum()

time\_t now = time(0)

DECLARE string oNum

DECLARE string stream strstream

DECLARE strstream now

DECLAREstrstream oNum

RETURN oNum

END FUNCTION

MAIN

Customer customer1[21]

IFstream customerFile

customerFile.open( customers.dat )

DECLARE string custNum

DECLARE string custName

DECLARE real getCredit

DECLARE string address

DECLARE string input[21]

DECLARE integereger position

DECLARE integereger position2

FOR (integer i=0i<21i++)

READIN file

custNum = StringHelper::parse(input[i],'|')[0]

custName = StringHelper::parse(input[i],'|')[1]

getCredit = stod(StringHelper::parse(input[i],'|')[2])

address= StringHelper::parse(input[i],'|')[3]

CALL customer1[i].setCustomerNum(custNum)

CALL customer1[i].setCustomerName(custName)

CALL customer1[i].setCredit(getCredit)

CALL customer1[i].setAddress(address)

ENDFOR

Address addy[21]

FOR (integer i=0i<21i++)

getline(customerFile, input[i])//reads in line from file

address= StringHelper::parse(input[i],'|')[3]

string streetAddress= StringHelper::parse(address,',')[0]

string zipcode= StringHelper::parse(address,',')[3]

CALL addy[i].setAddress(streetAddress)

CALL addy[i].setCity(city)

CALL addy[i].setState(state)

CALL addy[i].setZipCode(zipcodE)

ENDFOR

DECLARE IFstream inventoryFile

DECLARE Product product[100]

inventoryFile.open( inventory.dat )

DECLARE string read[100]

FOR (integer i=0i<20i++)

getline(inventoryFile, read[i])

DECLARE integereger itemNumber = stoi(StringHelper::parse(read[i],',')[0])

DECLARE string description =StringHelper::parse(read[i],',')[1]

DECLARE real price = stod(StringHelper::parse(read[i],',')[2])

DECLARE integereger stockQuantity= stoi(StringHelper::parse(read[i],',')[3])

CALL product[i].setItemNumber(itemNumber)

CALL product[i].setDescription(description)

CALL product[i].setPrice(price)

CALL product[i].setStockQuantity(stockQuantity)

ENDFOR

DECLARE string associate,customerName, buyItem

DECLARE integereger buyQty

DECLARE character option

OUTPUT Associate, please enter your name.

INPUT associate

OUTPUT Please enter the customers Name

INPUT customerName

DECLARE string x[21]//a

FOR(integer j=0j<21j++)

SET x[j]=customer1[j].getCustomerName()

ENDFOR

SET position = customerSearch(x,customerName,21)

DECLARE integereger buy

OUTPUT How many items would they like to purchase

INPUTbuy

DECLARE integer in[buy]

DECLARE integer qty

FOR(integer n=0n<buyn++)

OUTPUT Enter item number.

INPUTin[n]

ENDFOR

DECLARE integereger y[21]

FOR(integer j=0j<21j++)

SET y[j]=product[j].getItemNumber()

ENDFOR

DECLARE integer position1= checkItemNumber(y,in,21)

SET integer outItem=product[position1].getItemNumber()

OUTPUT Enter the quantity being purchased.

INPUT qty

integer z[20]

FOR (integer j=0j<20j++)

SET z[j]=product[j].getStockQuantity()

ENDFOR

position2=checkQty(z,qty,20)

SET integer oldQty=product[position2].getStockQuantity()

OUTPUT generateOrderNum

OUTPUT associate

OUTPUT customerName

OUTPUT Address: customer1[position].getAddress()

OUTPUT Item No ,Description ,Qty, Total

END MAIN