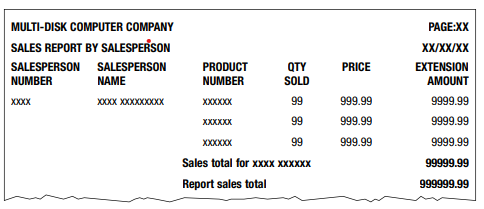
1. The Multi-Disk computer company requires a single-level control break program to produce a sales report by salesperson from their sales file. Design an algorithm that will read the sales file and create the sales report as shown below.

Each record on the sales file contains the salesperson’s number, name, the product number of the product sold, the quantity sold and the price of the product. There may be more than one record for each salesperson, depending on the products sold that month. The sales file has been sorted into ascending sequence of salesperson number.

Your program is to read the sales file sequentially, calculate the extension amount (price \* quantity sold) for each product sold and print a detail line for each record processed. Control total lines showing the sales total for each salesperson are to be printed on change of salesperson number. Print headings and column headings at the top of each page, allowing for 40 detail lines per page.

1. **Defining the Problem**

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Salesperson\_file | Read salesperson\_file | Headings |
| 1. Number | Print headings | Column\_headings |
| 1. Name | Print column headings | Sales\_detail |
| 1. Product\_number\_sold | Print sales\_detail | 1. Number |
| 1. Product\_quantity\_sold | Calculate extension\_amount | 1. Name |
| 1. Product\_price | Print extension\_amount | 1. Product\_number\_sold |
|  | Calculate salesperson\_total | 1. Product\_quantity\_sold |
|  | Calculate sales\_total | 1. Product\_price |
|  | Print salesperson\_total | salesperson\_total |
|  | Print sales\_total | Sales\_total |

1. **Group the activities into modules**

* perform\_initial\_processing
* A module to calculate extension amount
* A module to print salesperson detail
* A module to calculate extension\_amount
* A module to print salesperson\_detail
* A module to calculate salesperson\_total
* A module to calculate sales\_total
* A module to check\_detail\_line

1. **Hierarchy chart**

Salesperson\_file

perform\_initial\_processing

get\_data

Print\_salesperson\_detail

Calculate\_salesperson\_detail

Calculate\_sales\_total

Read sales

person\_file

check\_detail\_lines

Calculate\_extension\_amount

Read sales

person\_file

1. **Solution algorithm**

Sales\_report

1. perform\_initial\_processing
2. Print headings
3. Read first record
4. this\_control\_field = control field
5. prev\_control\_field = control field
6. print number, name
7. DOWHILE more records exist
8. get\_data
9. IF this\_control\_field NOT = prev\_control\_field THEN
10. Print column\_headings
11. calculate\_extension\_amount
12. print\_salesperson\_detail
13. prev\_control\_field = this\_control\_field
14. calculate\_sales\_total
15. calculate\_salesperson\_total
16. Print salesperson\_total
17. ELSE
18. calculate\_extension\_amount
19. Print product\_number\_sold, product\_quantity\_sold, product\_price, extension\_amount
20. prev\_control\_field = this\_control\_field
21. Calculate salesperson\_total
22. ENDIF
23. Calculate\_sales\_total
24. check\_detail\_lines
25. Read next record
26. this\_control\_field = control field
27. ENDDO
28. Print sales\_total

END

1. **Pseudocode module**

perform\_initial\_processing

1. Set extension\_amount to 0
2. Set salesperson\_total to 0
3. Set sales\_total to 0

END

get\_data

1. Get number, name, product\_number\_sold, product\_quantity\_sold, product\_price

END

calculate\_extension\_amount

1. Extension\_amount = product\_price \* product\_quantity\_sold

END

print\_salesperson\_detail

1. Print number, name, product\_number\_sold, product\_quantity\_sold, product\_price, extension\_amount

END

calculate\_salesperson\_total

1. Salesperson\_total = salesperson\_total + extension\_amount

END

Calculate\_sales\_total

1. Sales\_total = sales\_total + salesperson\_total

END

check\_detail\_lines

1. IF linecount > max\_detail\_lines THEN
2. Print headings
3. ENDIF

END

1. The Yummy Chocolates confectionery company requires a program to sequentially update its customer master file. A sequential file of update transactions is to be used as the input transaction file, along with the customer master file.

The customer master file contains the customer number, name, address (street, city, state and postcode) and account balance. The customer transaction file also contains these fields, as well as a transaction code of ‘A’ (add), ‘D’ (delete) and ‘U’ (update).

Both files have been sorted into customer number sequence. There can be multiple update transactions for any one customer master record and a new customer master file is to be created.

Transaction records are to be processed as follows:

1. If the transaction record is an ‘Add’, the transaction is to be written to the new customer master file.
2. If the transaction record is a ‘Delete’, the old master record with the same customer number is not to be written to the new customer master file.
3. If the transaction record is an update, the old master record with the same customer number is to be updated as follows:

if customer name is present, update customer name

if street is present, update street if town is present, update town

if state is present, update state

if postcode is present, update postcode

if balance paid is present, subtract balance paid from account balance on old customer master record.

As each transaction is processed, print the transaction details on the customer master audit report, with the message, ‘record added’, ‘record deleted’ or ‘record updated’ as applicable.

If a transaction record is in error, the transaction details are to be printed on the customer update errors report, with one of the following messages:

‘invalid addition, customer already exists’

‘invalid deletion, customer not on file’

‘invalid update, customer not on file’.

1. **Defining Diagram**

|  |  |  |
| --- | --- | --- |
| **input** | **process** | **Output** |
| Transaction\_file | Read transaction\_file |  |
| 1. Number | Read customer\_file |  |
| 1. Name | Validate code in transaction\_file | customer\_master\_audit |
| 1. Street | Add record | 1. Number |
| 1. City | Delete record | 1. Name |
| 1. Postcode | Update record | 1. Street |
| 1. Balance\_paid | Write customer\_master\_audit | 1. City |
| 1. code | Print transaction\_details | 1. Postcode |
| Customer\_file | Print transaction\_error | 1. balance |
| 1. Number |  | 1. Message |
| 1. Name |  | customer\_update\_error |
| 1. Street |  | 1. Number |
| 1. City |  | 1. Name |
| 1. Postcode |  | 1. Street |
| 1. balance |  | 1. City |
|  |  | 1. Postcode |
|  |  | 1. balance |
|  |  | 1. message |
|  |  | New\_customer\_file |
|  |  | 1. Number |
|  |  | 1. Name |
|  |  | 1. Street |
|  |  | 1. City |
|  |  | 1. Postcode |
|  |  | 1. balance |

1. **Modules**

* A module to search number
* A module to write customer\_update\_error
* A module to print customer\_update\_error
* A module to write new\_customer\_file
* A module to write customer\_master\_audit
* A module to print customer\_master\_audit
* A module to validate transaction\_file input

1. **Hierarchy Chart**

Transaction\_file

Search\_number

Write\_customer\_update\_error

Print\_customer\_update\_error

Write\_new\_customer\_file

Write\_customer\_master\_audit

Read sales

person\_file

Print\_customer\_master\_audit

Validate\_transaction\_file\_input

Customer\_file

1. **Solution Algorithm**

**Transaction\_record\_update**

1. Read customer\_file
2. Read transaction\_file
3. Copy customer\_file records to new\_customer\_file
4. DOWHILE record exists
5. GET number, name, street, city, postcode, balance from transaction\_file
6. IF code = “A” THEN
7. Search\_number
8. IF record exist THEN
9. Write\_customer\_update\_error
10. Customer\_update\_error message = “invalid addition, customer already exists”
11. Print\_customer\_update\_error
12. ELSE
13. Write\_new\_customer\_file
14. Write\_customer\_master\_audit
15. Customer\_master\_audit message = “record added”
16. Print\_customer\_master\_audit
17. ENDIF
18. ELSE IF code = “D” THEN
19. search\_number
20. IF record exist THEN
21. Write\_customer\_master\_audit
22. Customer\_master\_audit message = “record deleted”
23. Print\_customer\_master\_audit
24. ELSE
25. Write\_customer\_update\_error
26. Customer\_update\_error message = “invalid deletion, customer not on file”
27. Print\_customer\_update\_error
28. ENDIF
29. ELSE IF code = “U” THEN
30. Search\_number
31. IF record exist THEN
32. Validate\_transaction\_file\_input
33. Write\_new\_customer\_file
34. Write\_customer\_master\_audit
35. Customer\_master\_audit message = “record updated”
36. Print\_customer\_master\_audit
37. ELSE
38. Write\_customer\_update\_error
39. Customer\_update\_error message = “invalid update, customer not on file”
40. Print\_customer\_update\_error
41. ENDIF
42. ENDIF
43. ENDDO

END

1. **Pseudocode Module**

Search\_number

1. Search number from transaction\_file in customer\_file

END

Write\_customer\_update\_error

1. Write number, name, street, city, postcode, balance to customer\_update\_error

END

Print\_customer\_update\_error

1. Print number, name, street, city, postcode, balance, message from customer\_update\_error

END

Write\_new\_customer\_file

1. Write number, name, street, city, postcode, balance to new\_customer\_file

END

Write\_customer\_master\_audit

1. Write number, name, street, city, postcode, balance to customer\_master\_audit

END

Print\_customer\_master\_audit

1. Print number, name, street, city, postcode, balance, message from customer\_master\_audit

END

Validate\_transaction\_file\_input

1. IF name not null/empty THEN
2. Update name in new\_customer\_file
3. ENDIF
4. IF street not null/empty THEN
5. Update street in new\_customer\_file
6. ENDIF
7. IF city not null/empty THEN
8. Update city in new\_customer\_file
9. ENDIF
10. IF state not null/empty THEN
11. Update state in new\_customer\_file
12. ENDIF
13. IF postcode not null/empty THEN
14. Update postcode in new\_customer\_file
15. ENDIF
16. IF balance\_paid not null/empty THEN
17. Balance (in new\_customer\_file) = balance – balance\_paid
18. ENDIF

END