# Case 1: Take user input of customer data with validation

```
01 procedure TAKE_USER_INPUT_OF_CUSTOMER()
02     while TRUE do
03         print "Enter Customer ID (7-1000): "
04         customerId = READ_INPUT()
05         if customerId >= 7 and customerId <= 1000 then
06             break
07         else
08             print "Invalid ID. Please enter a number between 7 and 1000."
09         end if
10     end while
11
12     print "Enter First Name: "
13     firstName = READ_INPUT()
14     print "Enter Last Name: "
15     lastName = READ_INPUT()
16     print "Enter Phone: "
17     phone = READ_INPUT()
18
19     while TRUE do
20         print "Enter Service Cost (£20-300): "
21         serviceCost = READ_INPUT()
22         if serviceCost >= 20 and serviceCost <= 300 then
23             break
24         else
25             print "Invalid Service Cost. Please enter a number between £20 and £300."
26         end if
27     end while
28
29     print "Available Services:"
30     for each service in AVAILABLE_SERVICES do
31         print service
32     end for
33     print "Enter Services (comma-separated if multiple): "
34     services = READ_INPUT()
35
36     while TRUE do
```

```
37      print "Enter Stylist Name (Georgia, Richard, Bill): "
38      stylistName = READ_INPUT()
39      if stylistName == "Georgia" or stylistName == "Richard" or stylistName == "Bill" then
40          break
41      else
42          print "Invalid Stylist Name. Please enter Georgia, Richard, or Bill."
43      end if
44  end while
45
46  newCustomer = new Customer(customerId, firstName, lastName, phone, serviceCost, services,
stylistName, stylistId)
47  customerList.ADD(newCustomer)
48  originalCustomerList.ADD(newCustomer)
49  print "Customer added successfully!"
50 end procedure
```

# Case 2: Count Allocated Customer for Each Stylist

```
01 procedure COUNT_ALLOCATED_CUSTOMER_LIST()
02  stylistCustomerCount = new HASHTABLE()
03
04  for each customer in customerList do
05      stylistId = customer.GET_STYLIST_ID()
06      if stylistCustomerCount.CONTAINS(stylistId) then
07          stylistCustomerCount[stylistId] = stylistCustomerCount[stylistId] + 1
08      else
09          stylistCustomerCount[stylistId] = 1
10      end if
11  end for
12
13  print "Stylist ID    Stylist Name    Number of Customers"
14  for each stylist in stylistList do
15      count = stylistCustomerCount.GET_OR_DEFAULT(stylist.GET_STYLIST_ID(), 0)
16      print stylist.GET_STYLIST_ID(), stylist.GET_STYLIST_NAME(), count
17  end for
18 end procedure
```

# Case 3: Sort Highest Cost First

```
01 procedure QUICK_SORT_CUSTOMER_LIST()
02     QUICK_SORT(0, customerList.LENGTH() - 1)
03 end procedure
04
05 procedure QUICK_SORT(p, r)
06     if p < r then
07         q = PARTITION(p, r)
08         QUICK_SORT(p, q - 1)
09         QUICK_SORT(q + 1, r)
10     end if
11 end procedure
12
13 function PARTITION(p, r)
14     x = customerList[r].GET_SERVICE_COST() // pivot
15     i = p - 1
16     for j = p to r - 1 do
17         if customerList[j].GET_SERVICE_COST() > x then
18             i = i + 1
19             SWAP(customerList[i], customerList[j])
20         end if
21     end for
22     SWAP(customerList[i + 1], customerList[r])
23     return i + 1
24 end function
```

# Case 4: Sort Alphabetically All Customers by Their Last Name

```
01 QUICK_SORT_CUSTOMER_LIST_BY_LAST_NAME()
02     QUICK_SORT_BY_LAST_NAME(0, customerList.LENGTH() - 1)
03 procedure QUICK_SORT_BY_LAST_NAME(p, r)
04     if p < r then
```

```
05      q = PARTITION_BY_LAST_NAME(p, r)
06      QUICK_SORT_BY_LAST_NAME(p, q - 1)
07      QUICK_SORT_BY_LAST_NAME(q + 1, r)
08
09 PARTITION_BY_LAST_NAME(p, r)
10    x = customerList[r].GET_LAST_NAME() // pivot
11    i = p - 1
12    for j = p to r - 1 do
13       if customerList[j].GET_LAST_NAME() < x then
14          i = i + 1
15          Exchange customerList[i] with customerList[j]
16    Exchange customerList[i + 1] with  customerList[r]
17    return i + 1
```

# Case 5: Calculate Number of Customers and Total Amount Per Service

```
01 procedure CALCULATE_COST_OF_SERVICE()
02    serviceCustomerCount = new HASHTABLE()
03    serviceTotalCost = new HASHTABLE()
04
05    for each customer in customerList do
06       services = customer.GET_SERVICES().SPLIT(", ")
07       for each serviceName in services do
08          serviceCustomerCount[serviceName] =
serviceCustomerCount.GET_OR_DEFAULT(serviceName, 0) + 1
09          for each service in serviceList do
10             if service.GET_SERVICE_NAME() == serviceName then
11                serviceTotalCost[serviceName] = serviceTotalCost.GET_OR_DEFAULT(serviceName, 0.0) +
service.GET_PRICE()
12                break
13             end if
14          end for
15       end for
16    end for
17
18    print "Service              Number of Customers    Total Cost(£)"
```

```
    for each service in serviceList do
        serviceName = service.GET_SERVICE_NAME()
        count = serviceCustomerCount.GET_OR_DEFAULT(serviceName, 0)
        totalCost = serviceTotalCost.GET_OR_DEFAULT(serviceName, 0.0)
        print serviceName, count, totalCost
    end for
end procedure
```

# Case 6: Search Customer(s) Who Paid the Highest Service Cost

```
procedure DISPLAY_HIGHEST_SERVICE_COST_CUSTOMERS()
    if customerList.IS_EMPTY() then
        print "No customers available."
        return
    end if

    maxCost = customerTree.TREE_MAXIMUM(customerTree.ROOT).GET_SERVICE_COST()

    print "ID       First Name    Last Name    Phone       Service Cost(£)    Services            Stylist"
    for each customer in customerList do
        if customer.GET_SERVICE_COST() == maxCost then
            print customer.GET_CUSTOMER_ID(), customer.GET_FIRST_NAME(),
customer.GET_LAST_NAME(), customer.GET_PHONE(), customer.GET_SERVICE_COST(),
customer.GET_SERVICES(), customer.GET_STYLIST_NAME()
        end if
    end for
end procedure
```

# Case 7: Search Customer(s) Who Paid the Lowest Service Cost

```
procedure DISPLAY_LOWEST_SERVICE_COST_CUSTOMERS()
    if customerList.IS_EMPTY() then
```

```
03      print "No customers available."
04      return
05    end if
06
07    minCost = customerTree.TREE_MINIMUM(customerTree.ROOT).GET_SERVICE_COST()
08
09    print "ID      First Name    Last Name    Phone      Service Cost(£)    Services          Stylist"
10    for each customer in customerList do
11      if customer.GET_SERVICE_COST() == minCost then
12        print customer.GET_CUSTOMER_ID(), customer.GET_FIRST_NAME(),
customer.GET_LAST_NAME(), customer.GET_PHONE(), customer.GET_SERVICE_COST(),
customer.GET_SERVICES(), customer.GET_STYLIST_NAME()
13      end if
14    end for
15 end procedure
```

# Case 8: Search Customer(s) That Each Stylist Has

```
01 procedure SEARCH_CUSTOMERS_BY_STYLIST()
02    while TRUE do
03      print "Enter the stylist's name (Georgia, Richard, Bill): "
04      stylistName = READ_INPUT()
05      if stylistName == "Georgia" or stylistName == "Richard" or stylistName == "Bill" then
06        break
07      else
08        print "Invalid stylist name. Please enter Georgia, Richard, or Bill."
09      end if
10    end while
11
12    found = FALSE
13    print "ID      First Name    Last Name    Phone      Service Cost(£)    Services
Stylist"
14    for each customer in customerList do
15      if customer.GET_STYLIST_NAME() == stylistName then
16        print customer.GET_CUSTOMER_ID(), customer.GET_FIRST_NAME(),
customer.GET_LAST_NAME(), customer.GET_PHONE(), customer.GET_SERVICE_COST(),
customer.GET_SERVICES(), customer.GET_STYLIST_NAME()
17        found = TRUE
18      end if
```

```
19      end for
20
21      if not found then
22          print "No customers found for stylist, or this stylist does not exist: " + stylistName
23      end if
24 end procedure
```

# Case 9: Reset the Order of the Customer List

```
01 procedure RESET_CUSTOMER_LIST()
02      customerList.CLEAR()
03      customerList.ADD_ALL(originalCustomerList)
04 end procedure
```