# Assignment

## Algorithm, Flowchart – Catering System

Atreya Bain (Roll no. 24), Chirag Bapat(Roll no. 37)

Section: J

# Algorithm

## Catering system transactions

Problem definition: Design and develop an integrated solution of a caterer billing system to run a small scale business in a day to day event transaction activities. The solution provides complete details of the valid business details with user friendly environment along with the report details.

## Used variables and data structures:

Global variables:

1. cat_details : Contains a list of miscellaneous caterer details. Consists of:
    a. name (String) : String to store name of Catering company
    b. taxp (Float) : Tax Percentage
2. menu : Containing the menu list. Consists of:
    a. num_menu (Integer) : Number of items in the menu
    b. pieces (Array of custom structure) : Consists of number of each inventory item, which can be accessed by "pieces[i]" where i is the i-th element (Considering the array to be zero-indexed). Each of these items further consist of :-
        i. name (String) : Name of the food item
        ii. sprice (Float) : Selling price of the item
        iii. pcost (Float) : Production of the item
3. last_invoice : Stores the most recently generated/used invoice in the program. As at most one invoice is required to be loaded at once, only one such structure is used.
    a. recep (String) : Contains the name of the recipient the invoice is addressed to.
    b. item_numbers (Array of tuples): Each item in this tuple, represents the corresponding entry in the menu-list, and the corresponding quantity, where the former is stored, in the item's $0^{th}$ index.
    c. pieces_len (Integer) : Contains the number of items in
4. Invoice_list : Stores a list of names of generated bills to generate the reports from, and has:
    a. num_invoice (Integer) : Number of invoices generated so far
    b. invoice_name_list (Array of strings) : Contain the list of names of invoices generated so far

## Used subroutines

All variables have access to the above variables as they are global, and may or may not return a return a value to the main program, and subroutines returning a value maybe used to evaluate expressions, or can be stored into variables for future use.

The format for presenting a subroutine below is : <function_name>( [Argument type]:[Arguments] )

1. read_cat_det() : Load the caterer details file if present and return 1, else return 0
2. write cat_det() : Save the 'cat_details' structure in a file for reference.
3. print_cat_det() : Print out the caterer details in a formatted manner
4. read_menulist() : Load the menu list file if present and return 1, else return 0
5. write_menulist() : Save the 'menu' structure in a file for use.
6. print_menulist() : Print the menu list for reference
7. input_item_number() : Prints the menu, and returns the choice the user has submit
8. read_invoice_list() : Load from storage the list of invoices
9. write_invoice() : Make an invoice, and save it to storage, and update the invoice list
10. read_invoice() : Load an invoice from memory
11. print_invoice() : Print the invoice that has been loaded onto the memory.
12. report(): Generate report of number of items sold, and total sales, profit and tax.
13. check_substreq(String: string1, String: string2, Integer: position1, Integer: position2) : Checks if the strings are equal, between position1 and position 2 (including position1 and excluding position2)

Calls to the system to load a variable/structure state from storage has been denoted by the following functions:

1. FREAD( String: File name ) : Read a file name from memory
2. FWRITE( Variable/Data Structure, String: File name ) : Save to specified file path to storage
3. EXISTS(String: File name ) : Check if a given file exists or not, returns true (or 1) if exists, else returns false (or 0)
4. GET_DATETIME() : Returns the current system date and time in a string format

## Main Program:

```
Step 01:    START
Step 02:    flag=1
Step 03:    flag_hasloadedmenu = read_menulist()
Step 04:    flag_hasloadedcat = read_cat_det()
Step 05:    flag_hasloadedinvoicelist = read_invoice_list()
Step 06:    while ( flag )
Step 07:        PRINT “”
Step 08:        INPUT choice
Step 09:        If ( choice = 1 ) then
Step 10:            If ( flag_hasloadedcat = 1 ) then
Step 11:                write_cat_det()
Step 12:            else
Step 13:                PRINT “Caterer details not present”
Step 14:        Elseif ( choice = 2 ) then
Step 15:            flag_hasloadedmenu = read_menulist()
Step 16:            print_cat_det()
Step 17:        Elseif ( choice = 3 ) then
Step 18:            if ( flag_hasloadedmenu = 1 ) then
Step 19:                write_menulist()
Step 20:            else
Step 21:                PRINT “Menu List not present”
Step 22:        Elseif ( choice = 4 ) then
Step 23:            flag_hasloadedmenu = read_menulist()
Step 24:            print_menulist()
Step 25:        Elseif ( choice = 5 ) then
Step 26:            write_invoice()
Step 27:        Elseif ( choice = 6 ) then
Step 28:            read_invoice()
Step 29:            print_invoice()
Step 30:        Elseif ( choice = 7 ) then
Step 31:            report()
Step 32:        Else
Step 33:            PRINT “Invalid Choice”
Step 34:        Endif
Step 35:    End while
Step 36:    STOP
```

## Subroutines

### 1. read_cat_det() :

```
Step 01:     START
Step 02:     If( exists(“company.details”) ) then
Step 03:            cat_details = FREAD(“company.details”)
Step 04:            RETURN 1
Step 05:     Else
Step 06:            RETURN 0
Step 07:     Endif
```

### 2. write_cat_det()

```
Step 01:     START
Step 02:     PRINT “Enter company name”
Step 03:     INPUT cat_details.name
Step 04:     PRINT “Enter Tax%”
Step 05:     INPUT cat_details.taxp
Step 06:     FWRITE(cat_details,“company.details”)
Step 07:     RETURN
```

### 3. print_cat_det()

```
Step 01:     START
Step 02:     PRINT “Company Name: ”,cat_details.name
Step 03:     PRINT “Tax Percent: ” cat_details.taxp
Step 04:     RETURN
```

### 4. read_menulist()

```
Step 01:     START
Step 02:     If( exists(“menu.details”) ) then
Step 03:            menu = FREAD(“menu.details”)
Step 04:            RETURN 1
Step 05:     Else
Step 06:            RETURN 0
Step 07:     Endif
```

5. write_menulist()

```
Step 01:     START
Step 02:     PRINT “Number of item in the menu:”
Step 03:     INPUT menu.num_item
Step 04:     If (menu.num_menu>128) then
Step 05:            Goto Step 2
Step 06:     for I = 0 to num_item
Step 07:            PRINT “Enter item name”
Step 08:            INPUT menu.pieces[I].name
Step 09:            PRINT “Enter item price and production cost”
Step 10:            INPUT menu.pieces[I].sprice, menu.pieces[I].pcost
Step 11:     end for
Step 12:     FWRITE(menu,”menu.details”)
Step 13:     RETURN
```

6. print_menulist()

```
Step 01:     START
Step 02:     Print “Item Name    Price  Production”
Step 03:     for I = 0 to menu.num_menu
Step 04:            PRINT menu.pieces[i].name, menu.pieces[i].sprice, menu.pieces[i].pcost
Step 05:     end for
Step 06:     RETURN
```

7. input_item_number()

```
Step 01:     START
Step 02:     print_menulist()
Step 03:     INPUT choice
Step 04:     RETURN choice
```

8. read_invoice_list()

```
Step 01:     START
Step 02:     If( exists(“menu.details”) ) then
Step 03:            menu = FREAD(“menu.details”)
Step 04:            RETURN 1
Step 05:     Else
Step 06:            RETURN 0
Step 07:     Endif
```

9. write_invoice()

```
Step 01:     START
Step 02:     I = 0
Step 03:     filename = GET_DATETIME() + “.bill”
Step 04:     PRINT “Invoice Recepient:”
Step 05:     INPUT last_invoice.rep
Step 06:     PRINT “Enter item numbers
Step 07:     while( ( buffer = input_item_number() ) ≠ 0 )
Step 08:            last_invoice.item_numbers[I][0] = buffer-1
Step 09:            PRINT “Enter item quantity”
Step 10:            INPUT last_invoice.item_numbers[I][1]
Step 11:            I = I + 1
Step 12:     end while
Step 13:     PRINT “Number of items ordered:”,I-1
Step 14:     last_invoice.pieces_len = i-1
Step 15:     if ( last_invoice.pieces_len =0 )
Step 16:            RETURN 0;
Step 17:     endif
Step 18:     FWRITE(last_invoice,filename)
Step 19:     invoice_list.invoice_name_list[invoice_list.num_invoice] = filename
Step 20:     invoice_list.num_invoice = invoice_list.num_invoice + 1
Step 21:     FWRITE(invoice_list,”bill_list.details”)
Step 22:     RETURN last_invoice.pieces_len
```

10. read_invoice()

```
Step 01:     START
Step 02:     if (!exists("bill_list.details")) then
Step 03:            invoice_list.num_invoice = 0
Step 04:     else
Step 05:            read_invoice_list
Step 06:     endif
Step 07:     if (invoice_list.num_invoice = 0) then
Step 08:            PRINT "No invoices."
Step 09:     endif
Step 10:     PRINT "Choose invoice to print"
Step 11:     FOR I = 0 to invoice_list.num_invoice-1
Step 12:            PRINT I+1," ",invoice_list.invoice_name_list[I]
Step 13:     end for
Step 14:     INPUT CHOICE
Step 15:     filename = invoice_list.invoice_name_list[choice-1]
Step 16:     last_invoice = FREAD( filename )
Step 17:     RETURN
```

11. print_invoice()

```
Step 01:     START
Step 02:     PRINT last_invoice.recep
Step 03:     total_price = 0
Step 04:     PRINT "Items:"
Step 05:     for I = 0 to last_invoice.pieces_len
Step 06:            item_number = last_invoice.item_numbers[i][0]
Step 07:            item_quantity = last_invoice.item_numbers[i][1]
Step 08:            item_name = menu.pieces[ item_number ].name
Step 09:            item_price = menu.pieces[ item_number ].sprice
Step 10:            total_price = total_price + item_price*item_quantity
Step 11:            PRINT item_name,item_price,item_quantity
Step 12:     end for
Step 13:     PRINT "Billed Amount", total_price
Step 14:     RETURN
```

12. report()

```
Step 01:    START
Step 02:    fn = GET_DATETIME() + “.bill”
Step 03:    str_datebounds = {0,4,6,8}
Step 04:    num_items = 0
Step 05:    total_sprice = 0
Step 06:    total_profit = 0
Step 07:    total_pcost = 0
Step 08:    total_tax = 0
Step 09:    PRINT “Choose interval: 1. Year
            2. Month
            3. Day”
Step 10:    INPUT interval
Step 11:    for I = 0 to invoice_list.num_invoice
Step 12:        check =
check_substreq(invoice_list.invoice_name_list[i],fn,0,str_datebounds[choice])
Step 13:        if(check) then
Step 14:            last_invoice = FREAD( invoice_list.invoice_name_list[I] )
Step 15:            for J = 0 to last_invoice.pieces_len
Step 16:                item_number = last_invoice.item_numbers[i][0]
Step 17:                item_quantity = last_invoice.item_numbers[i][1]
Step 18:                item_number = last_invoice.item_numbers[i][0]
Step 19:                item_sprice = menu.pieces[ item_number ].sprice
Step 20:                item_pcost = menu.pieces[ item_number ].pcost
Step 21:                total_sprice = total_sprice + item_sprice * item_quantity
Step 22:                total_pcost = total_pcost + item_pcost * item_quantity
Step 23:                num_items = num_items + item_quantity
Step 24:            end for
Step 25:        end if
Step 26:    end for
Step 27:    if(num_items ≠ 0) then
Step 28:        total_tax = total_sprice * cat_details.taxp/100
Step 29:        total_profit = total_sprice + total_tax - total_pcost
Step 30:        PRINT “Totalling - ”
Step 31:        PRINT “Number of items: ”,num_items
Step 32:        PRINT “Total tax: ”,total_tax
Step 33:        PRINT “Total Production Costs: ”,total_pcost
Step 34:            PRINT “Total Sale Price”,total_sprice
Step 35:            PRINT “Total Profit”,total_profit
Step 36:     else
Step 37:            PRINT “No items”
Step 38:     endif
Step 39:     RETURN
```

13. check_substreq(string1, string2, Integer: position1, position2)

```
Step 01:     START
Step 02:     fl = 1
Step 03:     for I = position1 to position 2
Step 04:            if(string[1] = string2[i])
Step 05:                   fl = 0
Step 06:                   GOTO STEP 09
Step 07:            endif
Step 08:     end for
Step 09:     RETURN fl
```