**BACHELOR OF COMPUTER SCIENCE**
**SCHOOL OF COMPUTER SCIENCE**
**BINA NUSANTARA UNIVERSITY**
**JAKARTA**

# ASSESSMENT FORM

**Course: COMP6047001 - Algorithm and Programming**

**Method of Assessment: Case Study**

**Semester/Academic Year : 1/2022-2023**

**Name of Lecturer : Kanyadian Idananta, S.Kom., M.T.I.**

**Date : Rabu, 11 Januari 2023**

**Class : LD01**

**Topic : Material Review II**

| | |
|---|---|
| **Group Members :** | 1. Fendy Wijaya (2602092150)<br>2. –<br>3. –<br>4. –<br>5. –<br>6. –<br>7. –<br>8. – |

**Student Outcomes:**
(SO 2) Mampu merancang, mengimplementasikan, dan mengevaluasi solusi berbasis komputasi untuk memenuhi serangkaian persyaratan komputasi dalam konteks ilmu computer
*Able to design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of computer science.*

**Learning Objectives:**

(LObj 2.2) Mampu mengimplementasikan solusi berbasis komputasi untuk memenuhi serangkaian persyaratan komputasi tertentu dalam konteks ilmu komputer
*Able to implement a computing-based solution to meet a given set of computing requirements in the context of computer science*

| No | Assessment criteria | Weight | Excellent (85 - 100) | Good (75-84) | Average (65-74) | Poor (0 - 64) | Score | (Score x Weight) |
|---|---|---|---|---|---|---|---|---|
| 1 | Ability to identify the problems and explain the solution | **25%** | The problem is well defined and solution is clearly explained and detailed | The problem is well defined and solution is less clearly explained | The problem is defined and solution is less clearly explained | The problem and solution are badly defined and explained respectively | **100** | **25** |
| 2 | Apply logical thinking | **25%** | Correctly and effectively applying logic thinking to solve the problem. | Correctly applying logic thinking to solve the problem | Partially correct applying logic thinking to solve the problem | Incorrectly applied the logic thinking | **100** | **25** |
| 3 | Ability to construct a C program | **25%** | All the syntax in the program are correctly and effectively applied | All the syntax in the program are correctly applied | Only some of the syntax in the program are correctly applied | None of the syntax in the program are correctly applied | **100** | **25** |
| 4 | Ability to choose the appropriate algorithm | **25%** | Correctly choosing the most effective algorithm to solve the problem | Correctly choosing effective algorithm to solve the problem | Correctly choosing the algorithm to solve the problem | Incorrectly choosing the algorithm to solve the problem | **100** | **25** |
| | **Total Score:** ∑(Score x Weight) | | | | | | | **100** |

Remarks:

**ASSESSMENT METHOD**

Instructions

1. This is an individual assignment and will be held in review topic session with duration of 1 week, or week 13.
2. You will be given 3 questions. The questions are study case.
3. There will be 4 files that you need to upload (and zipped):
    a. One is a word file. This file consists of your explanation of solution. The explanation should be given using flowchart for better explanation. A correct and detailed flowchart will give higher score.
    b. Another 3 is .c files of your solution which will be checked by the respective lecturers.
4. Don't cheat.

**Note for Lecturers**:

1. This case study assignment will be held with duration of 1 week in review topic, or week 13.
2. The answer is manually checked by each lecturer (not by system).
3. You may refer to the rubric table given above.

# Questions

You are required to write a program that can read a file and perform several functions. The file needed can be downloaded from link: https://1drv.ms/u/s!AhuAx03LAKWtnOM9O1wlXSAR84Z67g?e=lVmH5x .

File description:

This file is a csv file; therefore, each column is separated with comma. This file consists of 3939 rows of Housing Data in Malaysia with no missing value for each row. This file also has a **header. Therefore, when your program loads the data, your program should be able to skip this header before passing the data into your record variable**. A glimpse of the data:

AoL > file.csv

```
1  Location 1,Location 2,Price,Rooms,Bathrooms,CarParks,Type,Area,Furnish
2  Mont-Kiara,Kuala-Lumpur,1000000,2,2,0,Built-up,1000,Partly
3  Cheras,Kuala-Lumpur,310000,3,2,0,Built-up,1000,Partly
4  Kepong,Kuala-Lumpur,358000,3,3,0,Built-up,1000,Partly
5  Taman-Desa,Kuala-Lumpur,455000,2,2,0,Built-up,1000,Partly
6  Kepong,Kuala-Lumpur,358000,3,3,0,Built-up,1000,Partly
7  Kepong,Kuala-Lumpur,358000,3,3,0,Built-up,1000,Partly
8  Bukit-Jalil,Kuala-Lumpur,505000,3,2,0,Built-up,1000,Partly
9  Jalan-Klang-Lama,Kuala-Lumpur,410000,3,2,0,Built-up,1000,Partly
10 Setapak,Kuala-Lumpur,278000,3,2,0,Built-up,1000,Partly
11 Sentul,Kuala-Lumpur,688000,3,2,0,Built-up,1000,Fully
12 Mont-Kiara,Kuala-Lumpur,660000,2,2,0,Built-up,1000,Fully
13 Jalan-Klang-Lama,Kuala-Lumpur,338000,3,2,0,Built-up,1000,Partly
```

You are required to perform 3 functions as follows:

1. **(40 Points) Describe**.

   This function explains the information from each column. When running this function, **give a prompt to get input from the user** as the name of the column you want to describe. Then, display:

   a. For column loc1, loc2, room, bathrooms, carparks, type, or furnish, display:

i. Frequency for each unique value
ii. Maximum frequency
iii. Minimum frequency

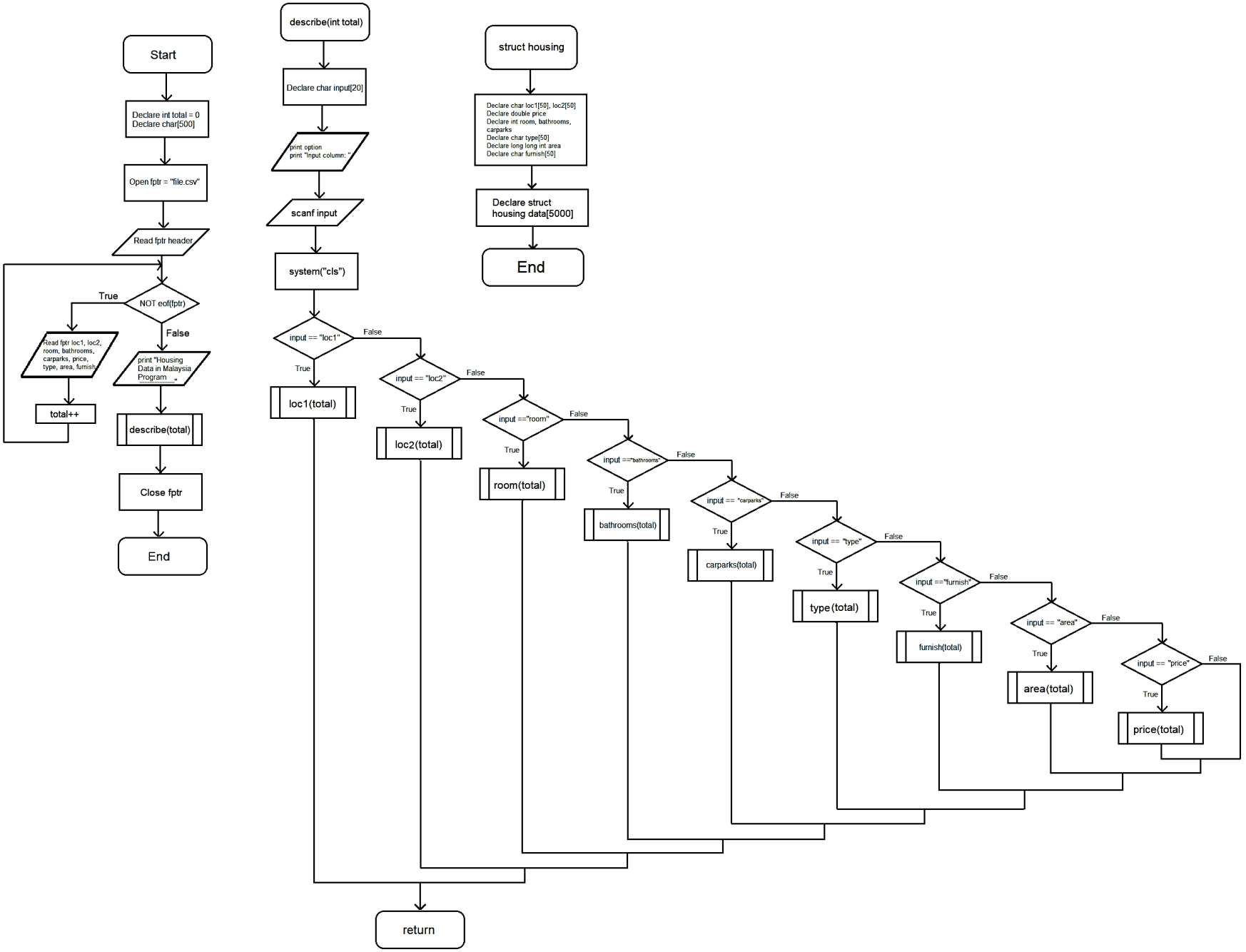
For example, if we call describe function followed with loc1 as column name, the program should display like this:

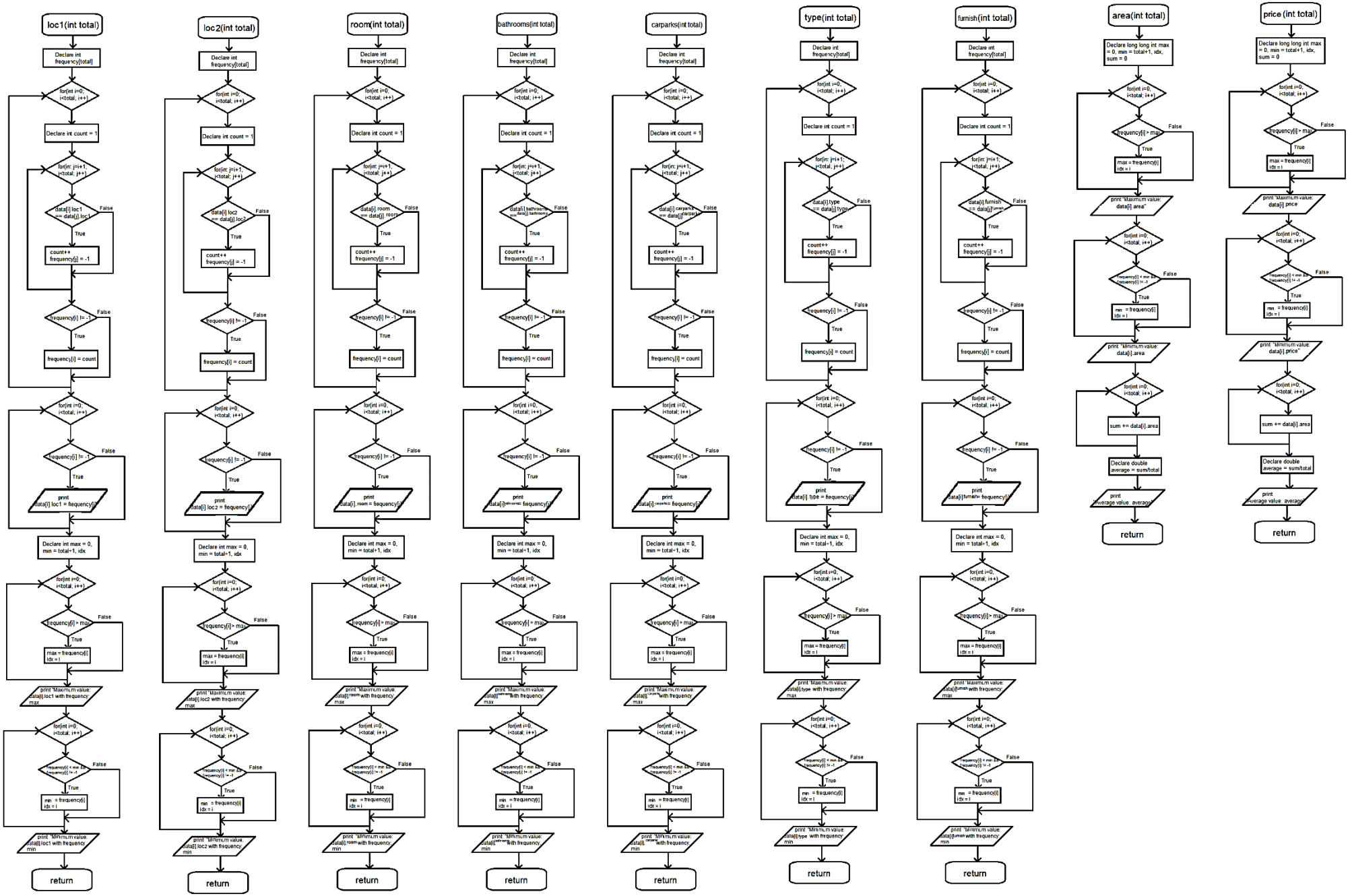
```
Batu-Caves : 19
Bangsar-South : 40
Kuchai-Lama : 37
Jinjang : 9
Bandar-Tasik-Selatan : 10
OUG : 40
Setiawangsa : 21
Sri-Hartamas : 63
Ampang-Hilir : 29
Seputeh : 21
Pandan-Indah : 10
Mid-Valley-City : 9
Brickfields : 19
Damansara : 5
Gombak : 3
Pandan-Jaya : 5
Alam-Damai : 1
Sunway-SPK : 8
Pandan-Perdana : 7
Other : 2
Happy-Garden : 1
Taman-Sri-Keramat : 1
TAMAN-MELATI : 1
Jalan-Sultan-Ismail : 12
Maximum value: Kepong with frequency: 450
Minimum value: TAMAN-MELATI with frequency: 1
```

b. For area and price column, display only:
    i. Minimum value
    ii. Maximum value
    iii. Average value

**Because area and price value are not discrete, therefore there isn't a need to perform frequency check for each of it.**

**Draw the Flowchart of your solution.**


Start
Declare int total = 0
Declare char[500]
Open fptr = "file.csv"
Read fptr header
NOT eof(fptr)
True
False
Read fptr loc1, loc2, room, bathrooms, carparks, price, type, area, furnish
total++
print "Housing Data in Malaysia Program"
describe(total)
Close fptr
End
describe(int total)
Declare char input[20]
print option
print "Input column: "
scanf input
system("cls")
input == "loc1"
loc1(total)
input == "loc2"
loc2(total)
input =="room"
room(total)
input =="bathrooms"
bathrooms(total)
input == "carparks"
carparks(total)
input == "type"
type(total)
input =="furnish"
furnish(total)
input == "area"
area(total)
input == "price"
price(total)
return
struct housing
Declare char loc1[50], loc2[50]
Declare double price
Declare int room, bathrooms, carparks
Declare char type[50]
Declare long long int area
Declare char furnish[50]
Declare struct housing data[5000]
End


loc1(int total)
Declare int frequency[total]
for(int i=0; i<total; i++)
Declare int count = 1
for(int j=i+1; j<total; j++)
data[j].loc1 == data[j].loc1
False
True
count++
frequency[j] = -1
frequency[i] != -1
frequency[i] = count
Declare int max = 0, min = total+1, idx
frequency[i] > max
max = frequency[i]
idx = i
print "Maximum value: data[i].loc1 with frequency: max
frequency[i] < min && frequency[i] != -1
min = frequency[i]
print "Minimum value: data[i].loc1 with frequency: min
return
loc2(int total)
data[i].loc2 == data[j].loc2
print "Maximum value: data[i].loc2 with frequency: max
print "Minimum value: data[i].loc2 with frequency: min
room(int total)
data[i].room == data[j].room
print "Maximum value: data[i].room with frequency: max
print "Minimum value: data[i].room with frequency: min
bathrooms(int total)
print "Maximum value: data[i].bathrooms with frequency: max
print "Minimum value: data[i].bathrooms with frequency: min
carparks(int total)
print "Maximum value: data[i].carparks with frequency: max
print "Minimum value: data[i].carparks with frequency: min
type(int total)
data[i].type == data[j].type
print "Maximum value: data[i].type with frequency: max
print "Minimum value: data[i].type with frequency: min
furnish(int total)
data[i].furnish == data[j].furnish
print "Maximum value: data[i].furnish with frequency: max
print "Minimum value: data[i].furnish with frequency: min
area(int total)
Declare long long int max = 0, min = total+1, idx, sum = 0
print "Maximum value: data[i].area"
sum += data[i].area
Declare double average = sum/total
print "Minimum value: data[].area"
print "Average value: average"
price (int total)
print "Maximum value: data[i].price"
print "Minimum value: data[].price"

2. **(40 Points) Search Data.**

   To be able to handle search function, ask user to give input with format:

   ```
   DataX in ColumnName
   ```

   Your program should be able to parse above input. **It is prohibited to use 3 string input**. Then, display all data that has that DataX.

   For example:

   ```
   Partly in furnish
   ```

   Above command will result in:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Kepong | Kuala-Lumpur | 357000 | 3 | 2 | 0 | Built-up | 973 | Partly |
| Salak-Selatan | Kuala-Lumpur | 248000 | 3 | 2 | 0 | Built-up | 973 | Partly |
| Mont-Kiara | Kuala-Lumpur | 1160000 | 2 | 2 | 0 | Built-up | 973 | Partly |
| City-Centre | Kuala-Lumpur | 800000 | 3 | 2 | 0 | Built-up | 975 | Partly |
| Sri-Petaling | Kuala-Lumpur | 390000 | 3 | 2 | 0 | Built-up | 975 | Partly |
| KLCC | Kuala-Lumpur | 1400000 | 2 | 2 | 0 | Built-up | 976 | Partly |
| KLCC | Kuala-Lumpur | 1400000 | 2 | 2 | 0 | Built-up | 976 | Partly |
| KLCC | Kuala-Lumpur | 1400000 | 2 | 2 | 0 | Built-up | 976 | Partly |
| KLCC | Kuala-Lumpur | 1400000 | 2 | 2 | 0 | Built-up | 976 | Partly |
| KLCC | Kuala-Lumpur | 1400000 | 2 | 2 | 0 | Built-up | 976 | Partly |
| Ampang | Kuala-Lumpur | 690822 | 2 | 2 | 0 | Built-up | 977 | Partly |
| Ampang | Kuala-Lumpur | 690822 | 2 | 2 | 0 | Built-up | 977 | Partly |
| Ampang | Kuala-Lumpur | 690822 | 2 | 2 | 0 | Built-up | 977 | Partly |
| Ampang | Kuala-Lumpur | 690822 | 2 | 2 | 0 | Built-up | 977 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 638000 | 3 | 2 | 0 | Built-up | 977 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 540000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 540000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Ipoh | Kuala-Lumpur | 486000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| Jalan-Klang-Lama | Kuala-Lumpur | 755000 | 3 | 2 | 0 | Built-up | 978 | Partly |
| KLCC | Kuala-Lumpur | 760000 | 2 | 1 | 0 | Built-up | 980 | Partly |

However, your program also should be able to search using only sub-string, for example:

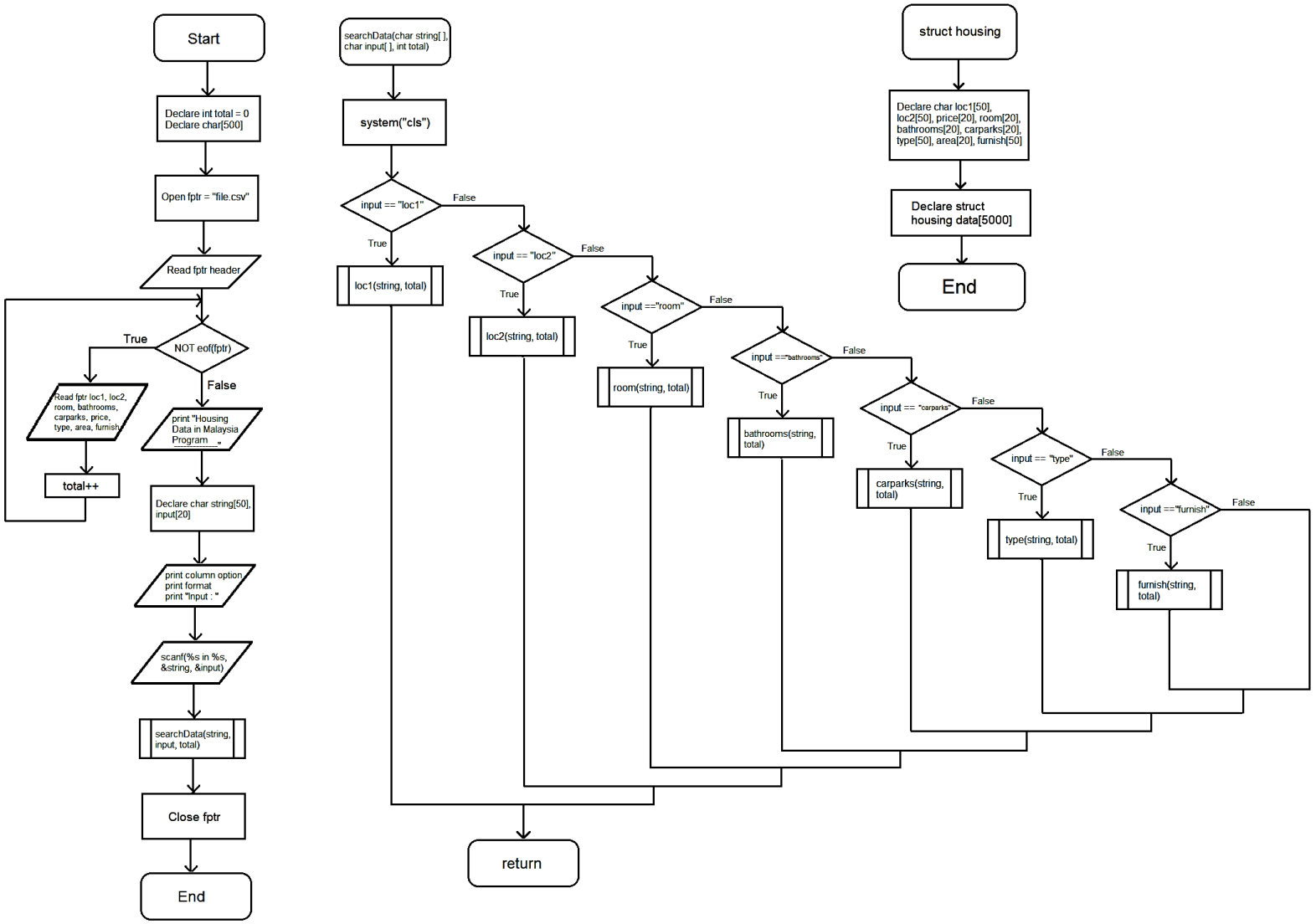
```
pong in loc1
```

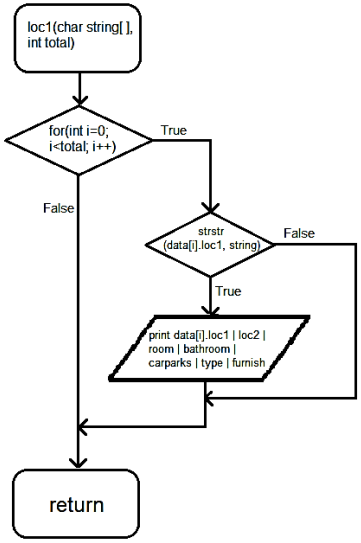
will result in:

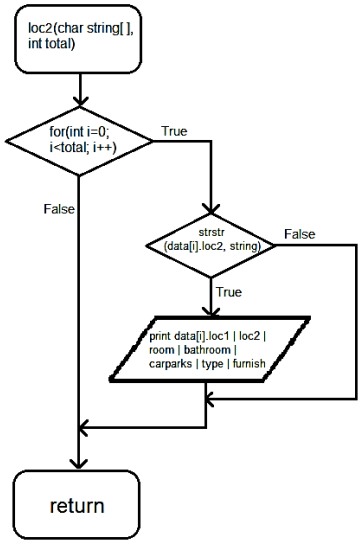
```
Kepong | Kuala-Lumpur | 2250000 | 5 | 4 | 0 | Land-area | 4500 | Partly
Kepong | Kuala-Lumpur | 930000  | 6 | 4 | 0 | Land-area | 4130 | Partly
Kepong | Kuala-Lumpur | 3400000 | 6 | 6 | 0 | Land-area | 4800 | Unfurnished
Kepong | Kuala-Lumpur | 398000  | 3 | 2 | 0 | Built-up | 630 | Partly
Kepong | Kuala-Lumpur | 465000  | 3 | 3 | 0 | Land-area | 630 | Partly
Kepong | Kuala-Lumpur | 365000  | 3 | 2 | 0 | Built-up | 630 | Partly
Kepong | Kuala-Lumpur | 480000  | 3 | 2 | 0 | Land-area | 630 | Partly
Kepong | Kuala-Lumpur | 350000  | 3 | 2 | 0 | Built-up | 630 | Partly
Kepong | Kuala-Lumpur | 480000  | 3 | 2 | 0 | Land-area | 630 | Partly
Kepong | Kuala-Lumpur | 4800000 | 5 | 6 | 0 | Land-area | 6466 | Partly
Kepong | Kuala-Lumpur | 200000  | 3 | 2 | 0 | Built-up | 650 | Partly
Kepong | Kuala-Lumpur | 200000  | 3 | 2 | 0 | Built-up | 650 | Partly
Kepong | Kuala-Lumpur | 185000  | 3 | 2 | 0 | Built-up | 650 | Unfurnished
Kepong | Kuala-Lumpur | 105000  | 3 | 2 | 0 | Built-up | 720 | Partly
Kepong | Kuala-Lumpur | 150000  | 3 | 2 | 0 | Land-area | 721 | Partly
Kepong | Kuala-Lumpur | 170000  | 3 | 2 | 0 | Built-up | 731 | Unfurnished
Kepong | Kuala-Lumpur | 218000  | 2 | 0 | 0 | Land-area | 750 | Unfurnished
Kepong | Kuala-Lumpur | 300000  | 3 | 2 | 0 | Built-up | 819 | Unfurnished
Kepong | Kuala-Lumpur | 218000  | 3 | 2 | 0 | Built-up | 841 | Partly
```

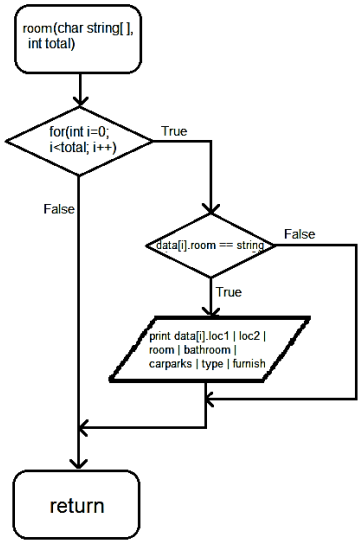
- **Give information if data searched doesn’t exist in the record.**
- **YOU ONLY HAVE TO DO SEARCH IN THE COLUMN OTHER THAN AREA AND PRICE.**

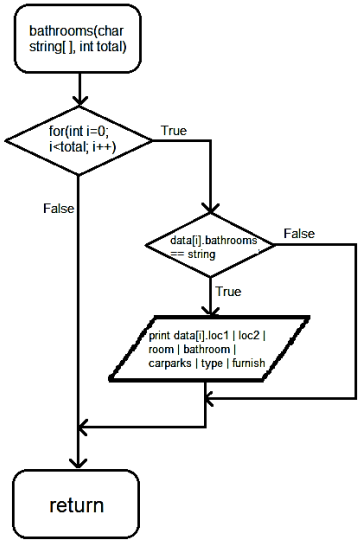
**Draw the Flowchart of your solution.**

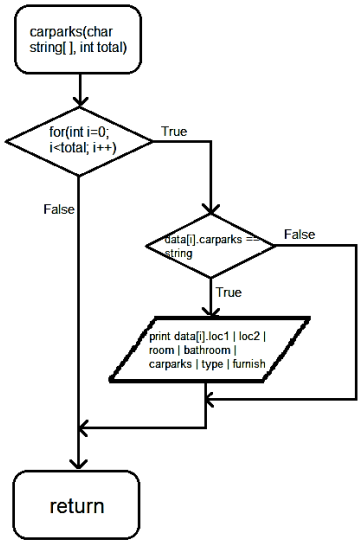

Start
Declare int total = 0
Declare char[500]
Open fptr = "file.csv"
Read fptr header
NOT eof(fptr)
True
False
Read fptr loc1, loc2, room, bathrooms, carparks, price, type, area, furnish
total++
print "Housing Data in Malaysia Program _"
Declare char string[50], input[20]
print column option
print format
print "Input : "
scanf(%s in %s, &string, &input)
searchData(string, input, total)
Close fptr
End
searchData(char string[ ], char input[ ], int total)
system("cls")
input == "loc1"
True
False
loc1(string, total)
input == "loc2"
loc2(string, total)
input =="room"
room(string, total)
input ==bathrooms"
bathrooms(string, total)
input == "carparks"
carparks(string, total)
input == "type"
type(string, total)
input =="furnish"
furnish(string, total)
return
struct housing
Declare char loc1[50], loc2[50], price[20], room[20], bathrooms[20], carparks[20], type[50], area[20], furnish[50]
Declare struct housing data[5000]
End

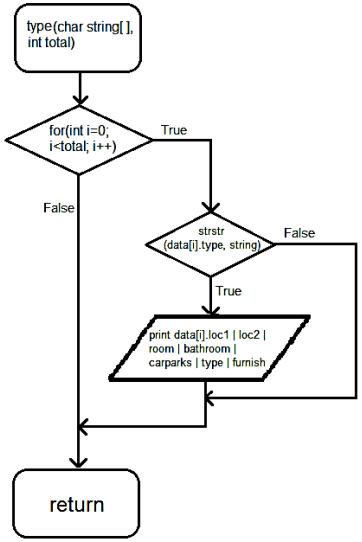

loc1(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
strstr (data[i].loc1, string)
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return

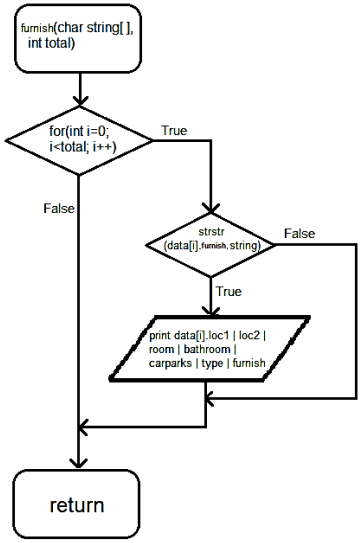

loc2(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
strstr (data[i].loc2, string)
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return


room(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
data[i].room == string
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return


bathrooms(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
data[i].bathrooms == string
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return

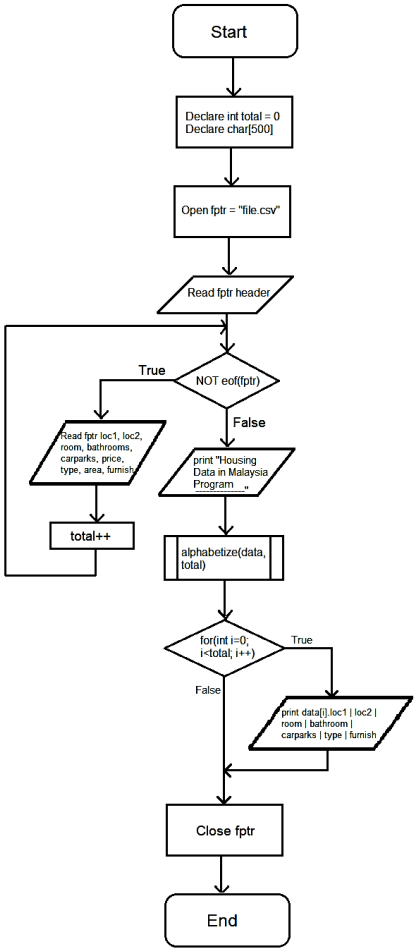

carparks(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
data[i].carparks == string
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return

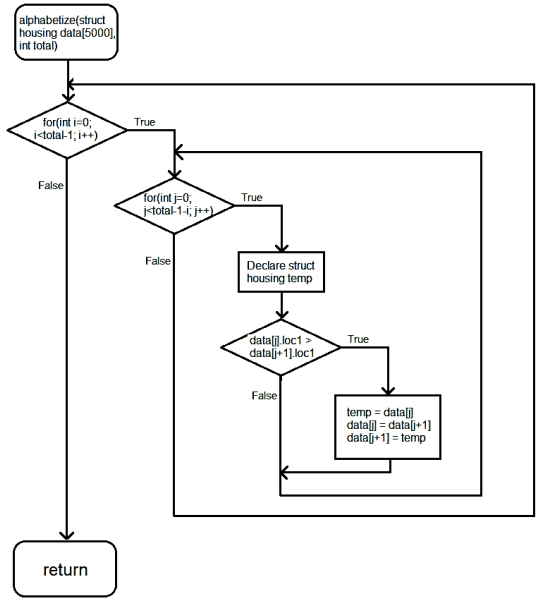

type(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
strstr (data[i].type, string)
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return

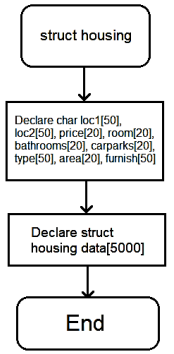

furnish(char string[ ], int total)
for(int i=0; i<total; i++)
True
False
strstr (data[i].furnish, string)
False
True
print data[i].loc1 | loc2 | room | bathroom | carparks | type | furnish
return

3. **(20 Point)** Please refer to **textbook Paul J. Deitel (2016). C how to program: with an introduction to C++, Chapter 8, Exercise 8.21 (page 404).** ***Alphabetizing a List of String.*** In the original question, you are asked to alphabetize list of string using 10 to 15 name of towns. **For this case, implement what is being asked using column Location 1.**

   **Draw the Flowchart of your solution.**

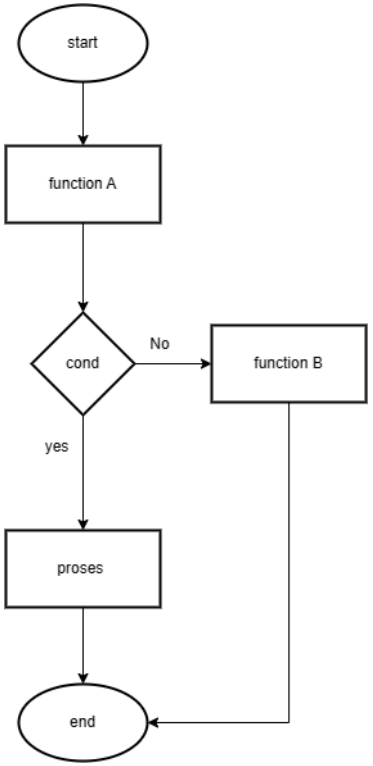


**Note:**

1. All solutions should be built using C code.
2. As mentioned in each question, you also need to give draw Flowcharts from your solution for each case number. If your solution contains more than 1 sub program, each sub program should also be drawn. For example:




Each Function A and Function B must have its own flowchart diagram.