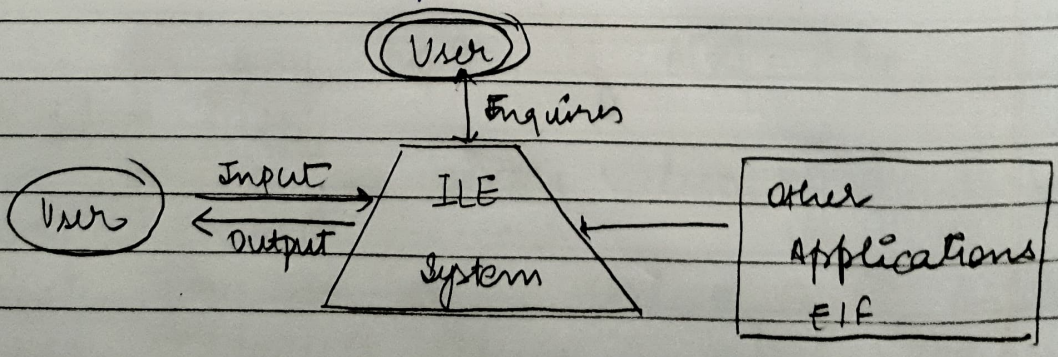
SET-A

NAME- ISHITA CHANDRA
SEMESTER- 3rd
PAPER CODE - BIT 203
DATE- 05/01/2022 11:00 AM

PROG.- BTECH, CSE
Roll- 08701012020
TITLE- Software Engineering

Ans No: (9). Function Count Method is used to calculate the functionality being delivered by an application. It measures the functionality from the point of view of the users.

It has a significant advantage over lines of code in the term that function count approach is independent of the language or methodologies or tools used for implementation. They make it possible to estimate development efforts in early phases of development. FPA makes it possible to compare the development productivity between different environments.



Here, ILF: Internal logical files
EIF: External Interfaces.

A system has 5 types of functional units
(a) Internal logical files
(b) External Interface files
(c) External inputs
(d) External outputs
(e) External Enquiries.

Internal logical files, external interface files are make up data function types and external inputs, external outputs external enquires make up transaction function types

These five parameters are known as information domain char and are assigned some weights.
The fn complexities are multiplied with the corresponding weights and values are added up to determine the unadjusted F of system.

For eg: To calculate functional point for the following data:

$$F\cdot P = \text{Count Total} \times CAF$$
$$\text{where } CAF = 0.65 + 0.01 \times \Sigma(f_i)$$

No. of user inputs = ~~27~~ 30
user outputs = 50
enquiries = 10
files = 5
External Interface = 1

6 & 8. ~~module weakness~~

Modularity is the concept of breaking a system into pieces called modules. Therefore, module is the part of a software system that can be seprately implemented.

~~Module~~ A program is a combination of various modules. Program weakness is therefore a useful measure. Program weakness depends on module weakness.

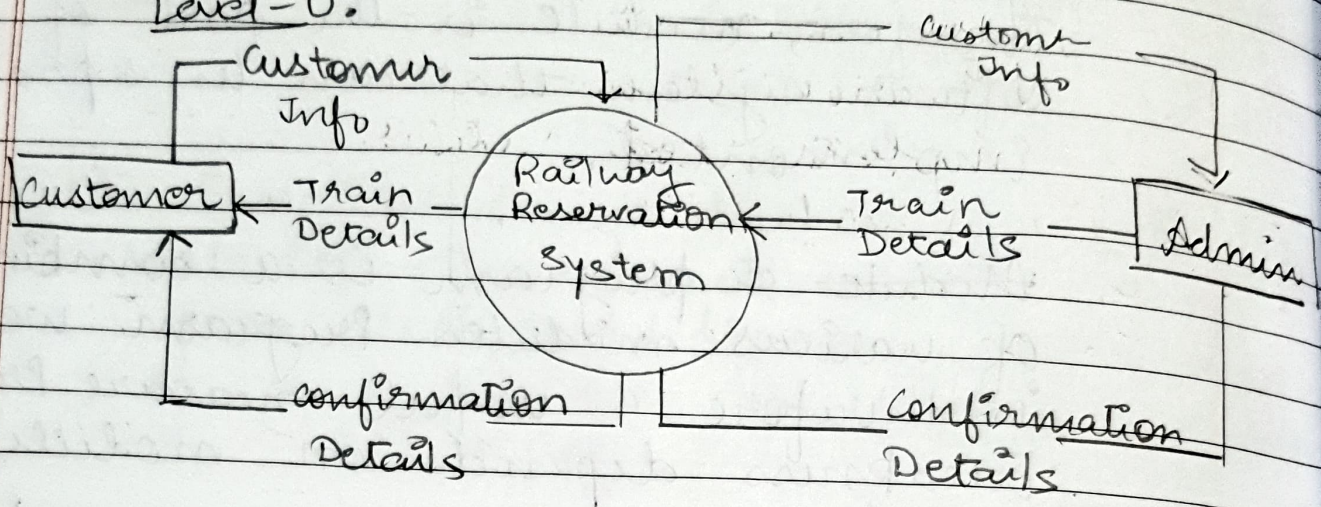
Module weakness has been defined as the product of average number of live variables ($\overline{LV}$) and average life variables (γ).

$\therefore WM = (\overline{LV} * \gamma)$.

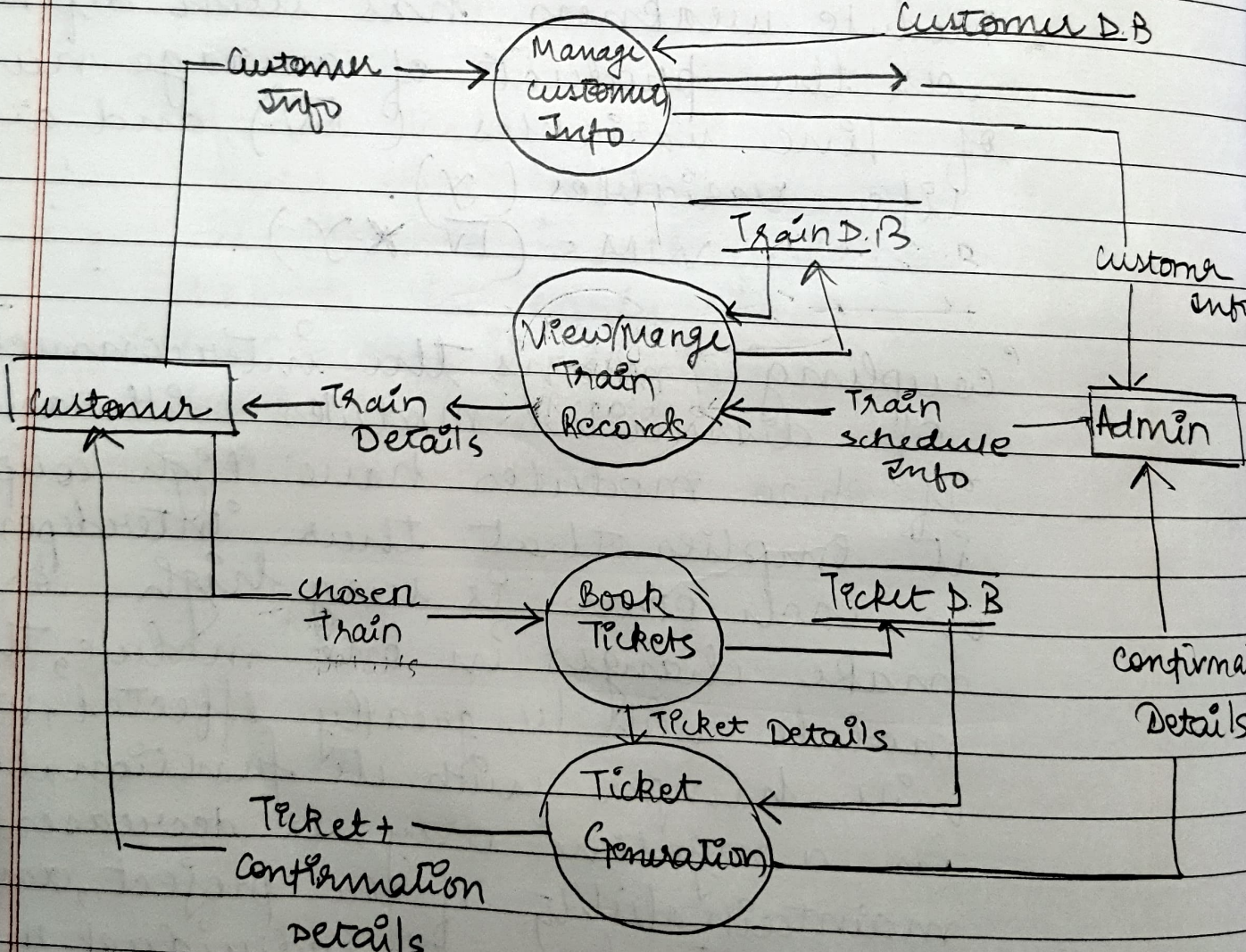
'coupling' means the interconnection of dissimilar modules with each other. If two modules have high coupling, it implies that their interdependence on each other is very high. If we make changes in one module, the other module will be greatly effected, which will hamper with its functionality in a negative way. It decreases the maintainability of the project, and the reusability factor of individual modules

Ans. No. 1 → (c) Railway Reservation System

Level-0:



Level-1:



2. $\lambda_0 = 10$ failures / cpu hr.
$\mu = 75$ failures
$\theta = 0.03$ / failures

failures experienced & failure intensity after 25 cup/hr

$$\mu(\tau) = \frac{1}{\theta} \ln(\lambda_0 \theta \tau + 1)$$

$$= \frac{1}{0.03} \ln(10 \times 0.03 + 25 \times 1) = 71 \text{ failures}$$

$$\lambda(\tau) = \lambda_0 / (\lambda_0 \theta \tau + 1)$$

$$= \frac{10}{(10 \times 0.03 \times 25 + 1)} = 1.176 \text{ failures/cpu hr.}$$

After 50 cpu hr.

$$\mu(\tau) = \frac{1}{\theta} \ln(\lambda_0 \theta \tau + 1)$$

$$= \frac{1}{0.03} \ln(10 \times 0.03 \times 50 + 1).$$

$$= 93 \text{ / failure}$$

$$\lambda(\tau) = \lambda_0 / (\lambda_0 \theta \tau + 1)$$

$$= \frac{10}{10 \times 0.03 \times 50 + 1} = 0.625 \text{ failures / cpu hr.}$$

PAGE No
DATE: / 201

087

Q2- b °.

```
1   #include <stdio.h>
2   #include <math.h>

3   int main{
4     double a, b, c, dis, r1, r2, realPart, imgPart;

5     printf("Enter coeff. a, b, & c: ");
6     scanf("%lf %lf %lf", &a, &b, &c);
7

8     dis = b*b - 4*a*c;
9
10    // cond^n for real and diff roots
11    if (dis > 0){

12      r1 = (-b + sqrt(dis))/(2*a);

13      r2 = (-b - Sqrt(dis))/(2*a);

14      printf("root 1 = %.2lf and root 2 =
          %.2lf", r1, r2);
15    }
16
17    // cond^n for real and equal roots.
18    else if (dis == 0){

19      r1 = r2 = -b/(2*a);
20      printf("root 1 = root 2 = %.2lf", r1);
21    }

22
23  // if roots aren't real
```

```
24  else {
25      realPart = -b/(2*a);
26      imgPart = sqrt(-dis)/(2*a);
27      printf("Root1 = %.2f + %.2fi and
            Root2 = %.2f - %.2fi", realPart,
            imgPart, realPart, imgPart);
28  }
29
30      return 0;
31  }
```

Number of distinct operands = 8
~~distinct operators~~ = 4

Helstead's :
n_1 = count of unique operator
n_2 = count " " operand
N_1 = " " total occu of operator
N_2 =

a	4	6	8	12	13	19	25	26
b	4	6	8	12	13	19	25	
c	4	6	8					
discriminant	4	8	11	12	13	18	26	
~~r1~~	4	12	14	19	20			
~~r2~~	4	13	14	19				
real part	4	25	27					
imag part	4	26	27					