PROGRAM - 3

Objective: Design the Constructive Cost Model to calculate effort and development time for organic, semidetached and embedded modes based on estimated size of the project

Theory:

COCOMO (Constructive Cost Model) is a regression model based on LOC, i.e. number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

The key parameters which define the quality of any software products, which are also an outcome of the COCOMO are primarily Effort & Schedule:

- Effort: Amount of labour that will be required to complete a task. It is measuredin person-months units.
- Schedule: Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

1. Organic – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

2. Semi-detached – A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. E.g.: Compilers or different Embedded Systems can be considered of Semi-Detached type.

3. Embedded – A software project with requiring the highest level of complexity, creativity, and experience requirement all under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

Algorithm:

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

$E_i=a*(KDLOC)b$

The value of the constant a and b are depends on the project type.

Basic COCOMO Model: The basic COCOMO model provides an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

Effort = $a_1*(KLOC)$ a_2 PM
Tdev = $b_1*(efforts)b_2$ Months

Where,

KLOC is the estimated size of the software product indicate in Kilo Lines of Code,

a_1, a_2, b_1, b_2 are constants for each group of software products,

Project	a_i	b_i	c_i	d_i
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Tdev is the estimated time to develop the software, expressed in months,

Effort is the total effort required to develop the software product, expressed in **person months (PMs)**.

Code:

```cpp
#include<bits/stdc
++.h>using
namespace std;

void calculate(float table[][4],char mode[][15], int size)
{
    int i;
    if(size>=2 &&
        size<=50)
        i=0;
    else if(size>50 &&
        size<=300)i=1;
    else
        if(s
        ize>
        300)
        i=2;
    cout<<"\nThe mode is "<<mode[i];
    float effort = table[i][0] *
    pow((400),table[i][1]);cout<<"\nEffort =
    "<<effort<<" Person-Month";
    float time = table[i][2] *
    pow((effort),table[i][3]);
    cout<<"\nDevelopment Time = "<<time<<"
    Months";

}


int main()
{
```

Output:

```
enter lines of code :
The mode is Semi-Detached
Effort = 2462.8 Person-Month
Development Time = 38.4539 Months
```

Results:

The experiment helped us in successfully implementing Basic COCOMO for effort and Development time.

Finding and learnings:

- From the development time versus the product size in KLOC, it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately. This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified.
- It is important to note that the effort and the duration estimations obtained using the COCOMO model are called a nominal effort estimate and nominal duration estimate. The term nominal implies that if anyone tries to complete the project in a time shorter than the estimated duration, then the cost will increase drastically.

PROGRAM - 4

Objective: Write a program to implement intermediate COCOMO model to estimate effort, development time and average staff size.

Theory:

The basic COCOMO model considers that the effort is only a function of the number of lines of code and some constants calculated according to the various software systems. The intermediate COCOMO model recognizes these facts and refines the initial estimates obtained through the basic COCOMO model by using a set of 15 cost drivers based on various attributes of software engineering.

Classification of Cost Drivers and their attributes:

1. **Product attributes -**

   - Required software reliability extent
   - Size of the application database
   - The complexity of the product

2. **Hardware attributes -**

   - Run-time performance constraints
   - Memory constraints
   - The volatility of the virtual machine environment
   - Required turnabout time

3. **Personnel attributes -**

   - Analyst capability
   - Software engineering capability
   - Applications experience
   - Virtual machine experience
   - Programming language experience

4. **Project attributes -**

   - Use of software tools
   - Application of software engineering methods
   - Required development schedule

$E = (a(KLOC)^b)*EAF$

EAF = Effort Adjustment factor

Code:

```cpp
#include<iostream>
#include<math.h>
using namespace std;
int fround(float x) {
    int a;
    x=x+0.5;
    a=x;
    return(a);
}
int main() {
    float
    table[3][4]={3.2,1.05,2.5,0.38,3.0,1.12,2.5,0.35,2.8,1.20,2.5,0.32};
    int i,j,size,model,rating;
    char mode[][15]={"Organic","Semi-Detached","Embedded"};
    char
    driver[15][6]={"RELY","DATA","CPLX","TIME","STOR","VIRT","TURN","ACAP","A
    EXP",
    "PCAP", "VEXP","LEXP","MODP","TOOL","SCED"};
    float effort,EAF,a,time,staff,productivity;
    float costdrivers[15][6]={  {0.75,0.88,1,1.15,1.40,-1},
                                {-1,0.94,1,1.08,1.16,-1},
                                {0.70,0.85,1,15,1.30,1.65},
                                {-1,-1,1,1.11,1.30,1.66},
                                {-1,-1,1,1.06,1.21,1.56},
                                {-1,0.87,1,1.15,1.30,-1},
                                {-1,0.87,1,1.07,1.15,-1},
                                {1.46,1.19,1,0.86,0.71,-1},
                                {1.29,1.13,1.00,0.91,0.82,-1},
                                {1.42,1.17,1,0.86,0.70,-1},
                                {1.21,1.10,1,0.90,-1,-1},
                                {1.14,1.07,1,0.95,-1,-1},
                                {1.24,1.10,1.00,0.91,0.82,-1},
                                {1.24,1.10,1,0.91,0.83,-1},
                                {1.23,1.08,1,1.04,1.10,-1} };
    printf("\nEnter the size of project : "); scanf("%d",&size);
    if(size>=2 && size<=50)
        model=0;
    else if(size>50 && size<=300)
        model=1;
    else if(size>300)
        model=2;
```

```c
    printf("\nMode =
%s",mode[model]);EAF=1;
    for(i=0;i
        <15;i
        ++){
        do{
        printf("\nRate cost driver %s on scale of 0-5 : ",driver[i]);
        printf("\n0-Very Low\t1-Low\t2-Nominal\t3-High\t4-Very High\t5-Extra
        High\n");scanf("%d",&rating);
        a=costdrivers[i]
        [rating];if(a==-
        1) {
            printf("\nNo value exist for this rating..Enter another
            rating...\n");
            }
        }
        while(a==-1);
            EAF=EAF*a;
    }
    printf("\nEAF = %f",EAF);
    effort=(table[model][0]*pow(size,table[model][1])) *
    EAF;time=table[model][2]*pow(effort,table[model][3]);
    staff=effort/time;
    productivity=size/effort;
```

Output:

```
Enter the size of project : 400

Mode = Embedded
Rate cost driver RELY on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver DATA on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver CPLX on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver TIME on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver STOR on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver VIRT on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver TURN on scale of 0-5 :
```

```
Rate cost driver TURN on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver ACAP on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver AEXP on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
0

Rate cost driver PCAP on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2

Rate cost driver VEXP on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2
```

```
Rate cost driver LEXP on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
3

Rate cost driver MODP on scale of 0-5 :
0-Very Low      1-Low    2-Nominal      3-High   4-Very High      5-Extra High
2
```

```
Rate cost driver TOOL on scale of 0-5 :
0-Very Low      1-Low    2-Nominal       3-High   4-Very High      5-Extra High
2

Rate cost driver SCED on scale of 0-5 :
0-Very Low      1-Low    2-Nominal       3-High   4-Very High      5-Extra High
2

EAF = 1.225500

Effort = 4549.288086 Person-Month
Development Time = 37.023876 Months
Average Staff Required = 123 Persons
Productivity = 0.087926 KLOC/Person-Month
PS C:\Users\Puneet Kumar\Desktop\cpp(CB)>
```

Results:

The experiment helped us in successfully implementing Intermediate COCOMO for effort and Development time.

Finding and learnings:

- From the development time versus the product size in KLOC, it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately. This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified.
- It is important to note that the effort and the duration estimations obtained using the COCOMO model are called a nominal effort estimate and nominal duration estimate. The term nominal implies that if anyone tries to complete the project in a time shorter than the estimated duration, then the cost will increase drastically.

PROGRAM - 5

Objective: Implement detailed COCOMO model to determine cost and schedule estimates for different phases.

Theory:

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed COCOMO, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort. The Six phases of detailed COCOMO are: Planning and requirements System design Detailed design Module code and test Integration and test Cost Constructive model the effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

Algorithm:

Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code KDLOC. Determine a set of 15 multiplying factors from various attributes of the project. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2. Calculate phase wise estimates by multiplying specified constants with calculated effort and development time

Code:

```cpp
#include<bits/
stdc++.h>using
namespace std;
int
fround(float
x){ int a;
x=x+0.5;
a
=
x
;
r
```

```cpp
    "VEXP","LEXP","MODP","TOOL","SCED"};
    float effort,EAF,a,time,staff,productivity;
    float costdrivers[15][6]={
        {0.75,0.88,1,1.15,1.40,-1},
        {-1,0.94,1,1.08,1.16,-1},
        {0.70,0.85,1,1.15,1.30,1.65},
        {-1,-1,1,1.11,1.30,1.66},
        {-1,-1,1,1.06,1.21,1.56},
        {-1,0.87,1,1.15,1.30,-1},
        {-1,0.87,1,1.07,1.15,-1},
        {1.46,1.19,1,0.86,0.71,-1},
        {1.29,1.13,1.00,0.91,0.82,-1},
        {1.42,1.17,1,0.86,0.70,-1},
        {1.21,1.10,1,0.90,-1,-1},
        {1.14,1.07,1,0.95,-1,-1},
        {1.24,1.10,1.00,0.91,0.82,-1},
        {1.24,1.10,1,0.91,0.83,-1},
        {1.23,1.08,1,1.04,1.10,-1}
    };
    float mp[6][5]={
        {0.06,0.16,0.26,0.42,0.16},
        {0.06,0.16,0.24,0.38,0.22},
        {0.07,0.17,0.25,0.33,0.25},
        {0.07,0.17,0.24,0.31,0.28},
        {0.08,0.18,0.25,0.26,0.31},
        {0.08,0.18,0.24,0.24,0.34}
    };
    float tp[6][5]={
        {0.10,0.19,0.24,0.39,0.18},
        {0.12,0.19,0.21,0.34,0.26},
        {0.20,0.26,0.21,0.27,0.26},
        {0.22,0.27,0.19,0.25,0.29},
        {0.36,0.36,0.18,0.18,0.28},
        {0.40,0.38,0.16,0.16,0.30}
    };
    char phases[5][30]={"Planning and Requirements","System Design","Detail Design","Mo
dule Code and Test","Integration and Test";
    printf("\nEnter the size of project : ");
    cin>>size;
    if(size>=2 && size<=50)
    model=0;
    else if(size>50 && size<=300)
    model=1;
    else if(size>300)
    model=2;
    printf("\nMode = %s",mode[model]);
```

```cpp
    EAF=1;
    for(i=0;i<15;i++)
    {
        do
        {
            printf("\nRate cost driver %s on scale of 0-5 : ",driver[i]);
            printf("\n0-Very Low\t1-Low\t2-Nominal\t3-High\t4-Very High\t5-
Extra High\n");
            cin>>rating;
            a=costdrivers[i][rating];
            if(a==-1)
            {
            printf("\nNo value exist for this rating..Enter another rating...\n");
            }
        }while(a==-1);
        EAF=EAF*a;
    }
    printf("\nEAF = %f",EAF);
    effort=(table[model][0]*pow(size,table[model][1])) * EAF;
    time=table[model][2]*pow(effort,table[model][3]);
    staff=effort/time;
    productivity=size/effort;
    if(model==0)
    {
        printf("\nEnter Ogranic - Small(0) or Medium(1) : ");
        cin>>S;
    }
    else if(model==1)
    {
        printf("\nEnter Semi-Detached - Medium(2) or Large(3) : ");
        cin>>S;
    }
    else if(model==2)
    {
        printf("\nEnter Embedded - Large(4) or Extra Large(5) : ");
        cin>>S;
    }
    printf("\n\nPhase-wise Distribution of Effort is :\n\n");
    for(i=0;i<5;i++)
    {
        printf("\n%s Phase = ",phases[i]);
        printf("%f",effort*mp[S][i]);
    }
    printf("\n\nPhase-wise Distribution of Development Time is :\n\n");
    for(i=0;i<5;i++)
    {
```

```
        printf("\n%s Phase =
        ",phases[i]);
        printf("%f",time*tp[S][i]
        );
    }
```

Output:

```
PS D:\bit stuffing> cmd /c .\"Intermediate.exe"          Rate cost driver VEXP on scale of 0-5 :
                                                         0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
Enter the size of project : 12                           3

Mode = Organic                                           Rate cost driver LEXP on scale of 0-5 :
Rate cost driver RELY on scale of 0-5 :                  0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High   3
3
                                                         Rate cost driver MODP on scale of 0-5 :
Rate cost driver DATA on scale of 0-5 :                  0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High   3
4
                                                         Rate cost driver TOOL on scale of 0-5 :
Rate cost driver CPLX on scale of 0-5 :                  0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High   3
3
                                                         Rate cost driver SCED on scale of 0-5 :
Rate cost driver TIME on scale of 0-5 :                  0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High   3
3
                                                         EAF = 1.018434
Rate cost driver STOR on scale of 0-5 :                  Enter Ogranic - Small(0) or Medium(1) : 1
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
2
                                                         Phase-wise Distribution of Effort is :
Rate cost driver VIRT on scale of 0-5 :
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
3                                                        Planning and Requirements Phase  =  2.709071
                                                         System Design Phase   =  7.224189
Rate cost driver TURN on scale of 0-5 :                  Detail Design Phase   =  10.836283
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High   Module Code and Test Phase   =  17.157448
3                                                        Integration and Test Phase   =  9.933268

Rate cost driver ACAP on scale of 0-5 :                  Phase-wise Distribution of Development Time is :
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
3
                                                         Planning and Requirements Phase  =  1.276132
Rate cost driver AEXP on scale of 0-5 :                  System Design Phase   =  2.020542
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High   Detail Design Phase   =  2.233230
3                                                        Module Code and Test Phase   =  3.615707
                                                         Integration and Test Phase   =  2.764952
Rate cost driver PCAP on scale of 0-5 :
0-Very Low    1-Low   2-Nominal    3-High  4-Very High    5-Extra High
3
```

Results:

The program helped us in successfully implementing Detailed COCOMO

Finding and learnings:

- From the development time versus the product size in KLOC, it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately. This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified.

- It is important to note that the effort and the duration estimations obtained using the COCOMO model are called a nominal effort estimate and nominal duration estimate. The term nominal implies that if anyone tries to complete the project in a time shorter than the estimated duration, then the cost will increase drastically.

Experiment - 6

AIM:

Write a program to implement Application Composition Estimation Model for effort estimation

THEORY:

In this model size is first estimated using Object Points. Object Points are easy to

identify and count. Object Points defines screen, reports, third generation (3GL)

modules as objects.

Object Point estimation is a new size estimation technique but it is well suited in

Application Composition Sector.

Estimation of Efforts:

Following steps are taken to estimate effort to develop a project
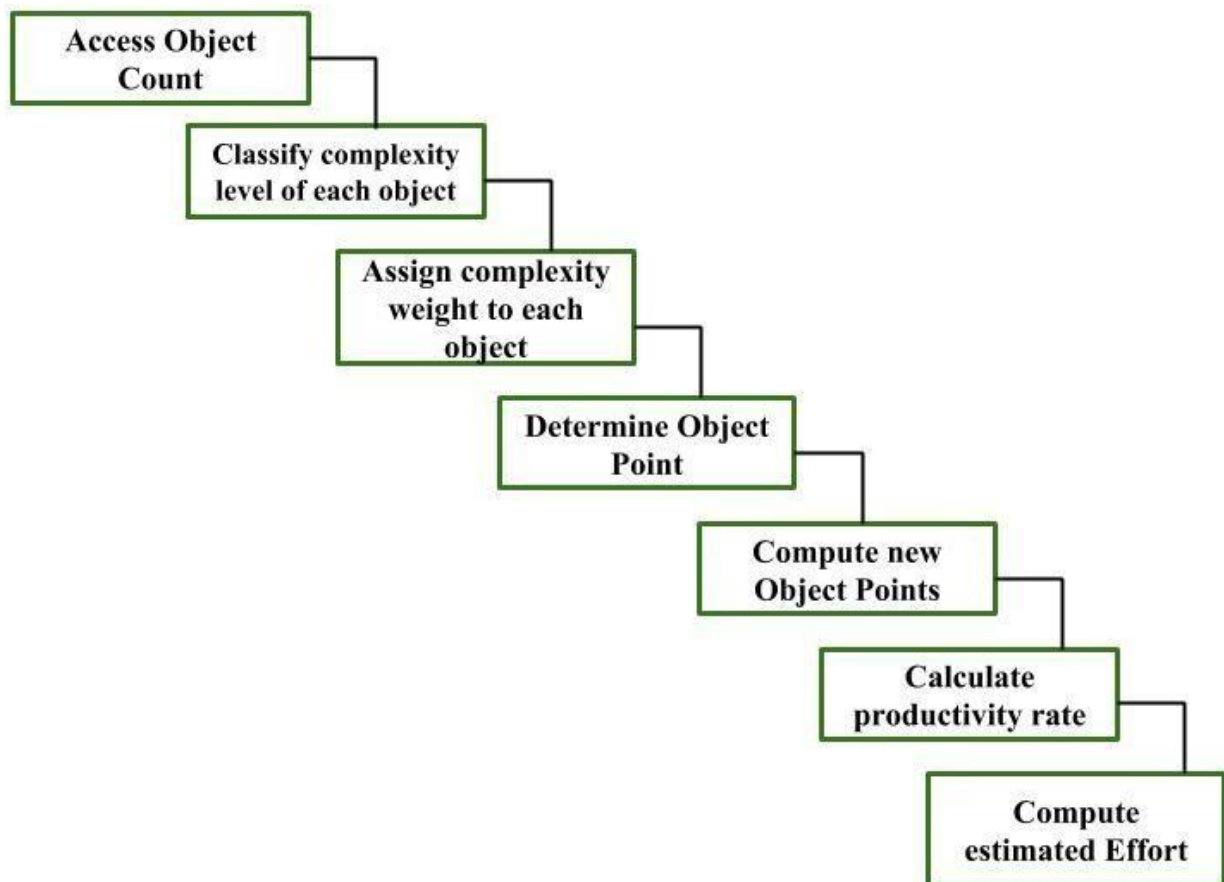
Step-1: Access Object counts

Estimate the number of screens, reports and 3GL components that will comprise this application.

Step-2: Classify complexity levels of each object

We have to classify each object instance into simple, medium and difficult complexity level depending on values of its characteristics.

Complexity levels are assigned according to the given table

No. of views contain	Sources of data tables		
	Total < 4 (< 2 servers < 3 clients)	Total < 8 (2 - 3 servers 3-5 clients)	Total 8 + (> 3 servers > 5 clients)
< 3	Simple	Simple	Medium
3 - 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

For Screens

No. of section contain	Sources of data tables		
	Total < 4 (< 2 servers < 3 clients)	Total < 8 (2 - 3 servers 3-5 clients)	Total 8 + (> 3 servers > 5 clients)
0 - 1	Simple	Simple	Medium
2 - 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

For Reports

Step-3: Assign complexity weights to each object

The weights are used for three object types i.e, screens, reports and 3GL components.

Complexity weight are assigned according to object's complexity level using following table

Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Components	-	-	10

Complexity Weight

Step-4: Determine Object Points

Add all the weighted object instances to get one number and this is known as object point count.

Object Point
= Sigma (number of object instances)
        * (Complexity weight of each object instance)

Step-5: Compute New Object Points (NOP)

We have to estimate the %reuse to be achieved in a project.

Depending on %reuse

NOP = [(object points) * (100 - %reuse)]/100

NOP are the object point that will need to be developed and differ from the object point count because there may be reuse of some object instance in project.

Step-6: Calculate Productivity rate (PROD)

Productivity rate is calculated on the basis of information given about developer's experience and capability.

For calculating it, we use following table

Developers experience & capability	Productivity (PROD)
Very Low	4
Low	7
Nominal	13
High	25
High	50

Productivity Rate

Step-7: Compute the estimated Effort

Effort to develop a project can be calculated as

Effort = NOP/PROD
Effort is measured in person-month.

CODE:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int screen[4], report[4];
    int comp_table[3][3] = {
        {0, 0, 1},
        {0, 1, 2},
        {1, 2, 2}};
    int weight_table[3][3] = {
        {1, 2, 3},
        {2, 5, 8},
        {-1, -1, 10}};
    int col, r_row, s_row, s_comp, r_comp;
```

```c
int rating, OP, reuse;
float NOP, effort;
int PROD[5] = {4, 7, 13, 25, 50};
printf("\nEnter number of Screens : ");
scanf("%d", &screen[0]);
printf("\nEnter number of Screen Views : ");
scanf("%d", &screen[1]);
printf("\nEnter number of clients and servers : ");
scanf("%d %d", &screen[2], &screen[3]);
printf("\nEnter number of Reports : ");
scanf("%d", &report[0]);
printf("\nEnter number of Report Sections : ");
scanf("%d", &report[1]);
printf("\nEnter number of clients and servers : ");
scanf("%d %d", &report[2], &report[3]);
if (screen[1] < 3)
    s_row = 0;
else if (screen[1] >= 3 && screen[1] <= 7)
    s_row = 1;
else if (screen[1] >= 8)
    s_row = 2;
if (report[1] == 0 || report[1] == 1)
    r_row = 0;

else if (report[1] == 2 || report[1] == 3)
    r_row = 1;
else if (report[1] >= 4)
    r_row = 2;
if (screen[2] < 2 && screen[3] < 3)
    col = 0;
else if ((screen[2] >= 2 && screen[2] <= 3) && (screen[3] >= 3 && screen[3] <= 5))
    col = 1;
else if (screen[2] > 3 && screen[3] > 5)
    col = 2;
s_comp = comp_table[s_row][col];
r_comp = comp_table[r_row][col];
```
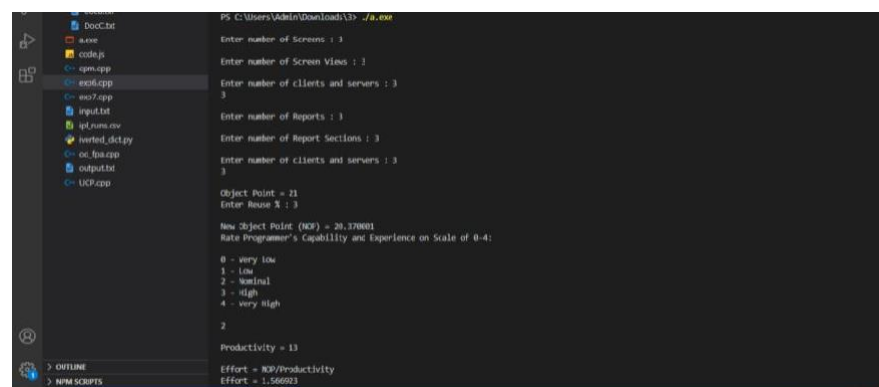
```
OP = (screen[0] * weight_table[0][s_comp]) + (report[0] * weight_table[1][r_comp]);
printf("\nObject Point = %d", OP);
printf("\nEnter Reuse % : ");
scanf("%d", &reuse);
NOP = (OP * (100 - reuse)) / 100.0;
printf("\nNew Object Point (NOP) = %f", NOP);
printf("\nRate Programmer's Capability and Experience on Scale of 0-4:\n");
printf("\n0 - Very Low\n1 - Low\n2 - Nominal\n3 - High\n4 - Very High\n\n");
scanf("%d", &rating);
printf("\nProductivity = %d", PROD[rating]);
effort = NOP / PROD[rating];
printf("\n\nEffort = NOP/Productivity");
printf("\nEffort = %f\n", effort);
}
```

OUTPUT:

RESULT:

We have successfully implemented Application Composition Estimation Model for effort estimation.

LEARNINGS AND FINDINGS:

COCOMO II is tuned to modern software life cycles. The original COCOMO model has been very successful, but it doesn't apply to newer software development practices as well as it does to traditional practices. COCOMO II targets modem software projects, and will continue to evolve over the next few years.

PROGRAM - 7

Objective: Write a program to implement Early Design Model and calculate the effort for the development of project

Theory:

The early design model uses Unadjusted Function Points (UFP) as measure of size. This model is used at the early stages of a software project when there is not enough information available about size of product which has t be developed, nature of target platform and nature of employees to be involved in development of projector detailed specifications of process to be used. This model can be used in Application Generator, System Integration, or Infrastructure Development Sector.

If B = 1.0, there is linear relationship between effort and size of product. If the value of B is not equal to 1, there will be non-linear relationship between size of product and effort. If B < 1.0, rate of increase of effort decreases as the size of product increases. If B > 1.0, rate of increase of effort increase as the size of product is increased.

This is due to growth of interpersonal communications and overheads due to growth of large system integration. Application composition model assumes value of B equal to 1. But Early Design Model assumes value of B to be greater than 1.

ALGORITHM:

Base equation used in COCOMO – II models is as follows – PMnominal =
A * (size)B

Where,
PMnominal = Effort for the project in person months
   A   = constant representing the nominal productivity where A = 2.5
   B   = Scale Factor
Size = size of the Software

The value of B can be calculated as -
B = 0.91 + 0.01 * (Sum of rang scaling factors for project)

Code:

```cpp
#include <bits/stdc++.h>
#define ll long long int
using namespace std;
void code()
{
    int size, i, rang;
    float PM_nominal, PM_adjusted;
    float B, F = 0, EM, a;
    char scale_factors[][30] = {"Precedentness", "Development Flexibility", "Risk Resol
uon", "Team  Cohesion", "Process Maturity"};
    float scale_table[5][6] = {
        {6.20, 4.96, 3.74, 2.48, 1.24},
        {5.07, 4.05, 3.04, 2.03, 1.01},
        {7.07, 5.65, 4.24, 2.83, 1.41},
        {5.48, 4.38, 3.29, 2.19, 1.10},
        {7.80, 6.24, 4.68, 3.12, 1.56}};
    string cost_drivers[7] = {"Product Reliability and Complexity(RCPX)", "Required  Re
usability(RUSE)", "Plaorm Difficulty(PDIF)", "Personnel Capability(PERS)", "Personnel
Experience(PREX)", "Facilies(FCIL)", "Required Development Schedule(SCED)"};
    float driver[7][7] = {
        {0.73, .81, .98, 1, 1.30, 1.74, 2.38},
        {-1, -1, .95, 1, 1.07, 1.15, 1.24},
        {-1, -1, .87, 1, 1.29, 1.81, 2.61},
        {2.12, 1.62, 1.26, 1, .83, .63, .50},
        {1.59, 1.33, 1.12, 1, .87, .71, .62},
        {1.43, 1.30, 1.10, 1, 1, .73, .62},
        {-1, 1.43, 1.14, 1, -1, 1, -1}};
    cout << "Enter the size of project (in KLOC) : ";
    scanf("%d", &size);
    cout << size << endl
         << endl;
    cout << "scale of scale_factors" << endl;
    cout << "0 - Very Low\n1 - Low\n2 - Nominal\n3 - High\n4 - Very High\n5 - Extra Hig
h\n\n";
    for (i = 0; i < 5; i++)
    {
        cout << "Rate Scaling Factor " << scale_factors[i] << " on the scale of 0-
5 : ";
        scanf("%d", &rang);
        cout << rang << endl;
        F = F + scale_table[i][rang];
    }
    B = (0.91 + (0.01 * F));
    printf("\nB (scale factor) = %f", B);
    PM_nominal = 2.5 * pow(size, B);
    printf("\nNominal Effort = %f Person-Month", PM_nominal);
```

```cpp
    cout << endl
        <
    <
    endl
    ;EM
    = 1;
    cout << "scale of cost_drivers" << endl;
    cout << "0 - Extra Low\n1 - Very Low\n2 - Low\n3 - Nominal\n4 - High\n5 -
    Very High
\n6 - Extra  High\n";
    for (i = 0; i < 7; i++)
    {
        do
        {
            cout << endl;
            cout << "Rate Cost Driver " << cost_drivers[i] << " on scale of
            (0-6) : ";scanf("%d", &rang);
            cout << rang;
            a =
            driver[i][r
            ang];if (a
            == -1)
                printf("\nNo value exists for this Rang..Enter another
                rang..\n");
        } while
        (a == -
        1);EM =
```

Output:

```
Enter the size of project (in KLOC) : 200
200

scale of scale_factors
0 - Very Low
1 - Low
2 - Nominal
3 - High
4 - Very High
5 - Extra High
```

```
Rate Scaling Factor Precedentness on the scale of 0-5 : 2
Rate Scaling Factor Development Flexibility on the scale of 0-5 : 2
Rate Scaling Factor Risk Resoluon on the scale of 0-5 : 2
Rate Scaling Factor Team  Cohesion on the scale of 0-5 : 2
Rate Scaling Factor Process Maturity on the scale of 0-5 : 2

B (scale factor) = 1.099900
Nominal Effort = 848.873413 Person-Month

scale of cost_drivers
0 - Extra Low
1 - Very Low
2 - Low
3 - Nominal
4 - High
5 - Very High
6 - Extra  High

Rate Cost Driver Product Reliability and Complexity(RCPX) on scale of (0-6) : 3

Rate Cost Driver Required  Reusability(RUSE) on scale of (0-6) : 3

Rate Cost Driver Plaorm Difficulty(PDIF) on scale of (0-6) : 3

Rate Cost Driver Personnel Capability(PERS) on scale of (0-6) : 3

Rate Cost Driver Personnel  Experience(PREX) on scale of (0-6) : 3

Rate Cost Driver Facilies(FCIL) on scale of (0-6) : 3

Rate Cost Driver Required Development Schedule(SCED) on scale of (0-6) : 3



Adjusted Effort (PM_adjusted) = 848.873413 Person-Month
C:\Users\Nitin Chawla\Desktop\coding>_
```

Results:

The experiment helped us in successfully implement Early Design Model for effort estimation.