

**PPG COLLEGE OF ARTS AND SCIENCE**  
(Affiliated to Bharathiar University)  
Saravanampatti, Coimbatore - 641035

**DEPARTMENT OF COMMERCE WITH PA**



**PRACTICAL RECORD**

**SUBJECT : ORACLE & C++**

<b>NAME</b>	
<b>REGISTER NO</b>	
<b>CLASS</b>	<b>II B.Com PA</b>
<b>BATCH</b>	<b>2018-2021</b>

**PPG COLLEGE OF ARTS AND SCIENCE**  
(Affiliated to Bharathiar University)  
Saravanampatti, Coimbatore - 641035

**Certified as a Bonafide record of work done by**

**Mr/Ms.** \_\_\_\_\_

**Reg. No:** \_\_\_\_\_

**of II B.Com PA during the year 2019-2020**

**Staff-In Charge**

**Head of the Department**

**Submitted for the Practical Examination held on \_\_\_\_\_ at**

**PPG College of Arts and Science, Coimbatore – 641035.**

**Internal Examiner**

**External Examiner**

## TABLE OF CONTENT

### ORACLE LAB

S. NO.	DATE	TITLE	PAGE NO.	SIGNATURE
1		COMPANY DETAILS		
2		EMPLOYEE DETAILS		
3		STUDENTS DETAILS		
4		PRODUCT DETAILS		
5		PAYROLL DETAILS		
6		BOOK DETAILS		
7		BANK DETAILS		

## TABLE OF CONTENT

### C++ LAB

S. NO.	DATE	TITLE	PAGE NO.	SIGNATURE
1		STRAIGHT LINE METHOD		
2		DIMINISHING BALANCE METHOD		
3		ECONOMIC ORDER QUANTITY		
4		PAYROLL MANAGEMENT		
5		SIMPLE AND COMPOUND INTEREST		
6		NET INCOME OF A FAMILY		
7		LIBRARY BOOK DETAILS		
8		COST SHEET		
9		MARGIN OF SAFETY		
10		BANK TRANSACTION		
11		WORKING CAPITAL		
12		STUDENTS MARK STATEMENT		

ORACLE

EX NO: 1	<b>COMPANY DETAILS</b>
DATE	

**AIM:**

To create a table named “company” with various fields such as company name, Proprietor, address, supplier name, employee and gp percent for 10 employees.

**ALGORITHM:**

**Step 1:** Start The Program.

**Step 2:** Create a table with the name Company Using Create Table Query.

**Step 3:** Insert the employee details as records In the Company table using insert query.

**Step 4:** Display the records in the company table in ascending order.

**Step 5:** Use select query to display the company name as ‘Telco’.

**Step 6:** Use the select and order by query to display the GP percent greater than 20 and in ascending order.

**Step 7:** Display the records of the employee name from 300 to 1000.

**Step 8:** Stop the process.

## SQL Queries and Output

```
SQL> create table company(company_name char(10),proprietor char(10),address  
char(10),supplier_name char(10),no_of_employees char(4),gp_percent number(6,4));
```

Table created.

```
SQL> insert into company values('kamal','karthi','chennai','kumar',100,20.9);
```

1 row created.

```
SQL> insert into company values('kanan','kani','karur','saran',100,30.9);
```

1 row created.

**A) Display all the records of the company which are in the ascending order of GP\_PERCENT percent.**

```
SQL> select * from company ORDER BY gp_percent percent;
```

COMPANY_NA	PROPRIETOR	ADDRESS	SUPPLIER_N	NO_O	GP_PERCENT
-----	-----	-----	-----	----	-----
Saraa	Krishna	annur	sindhu	200	10.9
Kamal	karthi	Chennai	kumar	100	20.9
Ketthu	varun	Coimbatore	ramu	130	30.7
Kanan	kani	karur	saran	100	30.9
Hamsha	vasanth	trichy	raja	750	40.7
Sandhiya	vicky	Coimbatore	telco	500	40.9
Hari	vasanthi	tirupur	tata	450	40.9
Ragul	varjana	Chennai	kamal	530	50.7
Hoffee	anbu	trichy	ram	100	50.9
Arun	arun	trichy	telco	600	60.9

10 rows selected.

**B) Display the name of the company whose supplier name is “telco”.**

SQL> select company\_name from company WHERE supplier\_name='telco';

COMPANY\_NAME

-----

sandhiya

arun

**C) Display the details of the company whose GP percentage is greater than 20 and order by gp percent.**

SQL> select \* from company WHERE gp\_percent>20 order by gp\_percent;

COMPANY_NAME	PROPRIETOR	ADDRESS	SUPPLIER_NAME	NO_OF_EMPLOYEES	GP_PERCENT
--------------	------------	---------	---------------	-----------------	------------

-----

-----

-----

-----

-----

-----

Kanan	kani	karur	saran	100	30.9
-------	------	-------	-------	-----	------

Sandhiya	vicky	Coimbatore	telco	500	40.9
----------	-------	------------	-------	-----	------

2 rows selected.

**D) Display the detail of the company having the employee ranging from 300 to 1000.**

SQL> select \* from company WHERE NO\_OF\_EMPLOYEES BETWEEN 300 and 1000;

COMPANY_NAME	PROPRIETOR	ADDRESS	SUPPLIER_NAME	NO_OF_EMPLOYEES	GP_PERCENT
--------------	------------	---------	---------------	-----------------	------------

-----

-----

-----

-----

-----

-----

Sandhiya	Vicky	coimbatore	telco	500	40.9
----------	-------	------------	-------	-----	------

Vicky	arun	trichy	telco	600	60.9
-------	------	--------	-------	-----	------

Hoffee	anbu	trichy	ram	1000	50.9
--------	------	--------	-----	------	------

Hari	vasanthi	tirupur	tata	450	40.9
------	----------	---------	------	-----	------

Hamsha	vasanth	trichy	raja	750	40.7
--------	---------	--------	------	-----	------

Ragul	varjana	Chennai	kamal	530	50.7
-------	---------	---------	-------	-----	------

6 rows selected.



**E) Display the name of the company whose supplier is same as the tata's.**

SQL> SELECT company\_name from company WHERE supplier\_name LIKE '%tata';

COMPANY\_NAME

-----

hari

**RESULT**

EX NO: 2	<b>EMPLOYEE DETAILS</b>
DATE	

**AIM:**

To create a table named “employee” with the various fields as name, code, address, designation, grade, date of join and salary.

**ALGORITHM:**

**Step 1:** start the program.

**Step 2:** Create the employee table using the create table query.

**Step 3:** Insert the records in the employee table using insert query.

**Step 4:** Display the records in the salary greater than Rs.10000.

**Step 5:** By using order by query display the record in ascending order according to the employee code.

**Step 6:** Display the total salary of the employee where grade is ‘A’ using sum () function.

**Step 7:** By using a select query display the names of the employees who earn more than Ravi.

**Step 8:** Stop the process.

## SQL Queries and Output

```
SQL> create table employee (employee_name varchar2(15), employee_code number(6),  
address varchar2(10), designation varchar2(15), grade varchar2(1), date_of_join date  
, salary number(10,2));
```

Table created.

```
SQL> insert into employee values ('hari',1000,'salem','manager','j','1/may/2000  
,8000.00);
```

1 row created.

```
SQL> insert into employee values ('mani',2000,'coimbatore','chairman','a  
n/2000',10000.00);
```

1 row created.

```
SQL> select * from employee;
```

EMPLOYEE_NAME	EMPLOYEE_CODE	ADDRESS	DESIGNATION	G	DATE_OF_J	SALARY
-----	-----	-----	-----	-	-----	-----
hari	1000	salem	manager	j	01-MAY-00	8000
mani	2000	coimbatore	chairman	a	01-JAN-00	10000
raj	200	karur	accountant	c	02-FEB-00	5000
kumar	300	tirupur	manager	d	01-MAR-00	12000
kaml	400	chennai	president	e	02-MAY-00	9000
kannan	500	madurai	manager	g	02-FEB-00	5000
tamil	600	madurai	manager	r	02-MAR-00	7000
suba	800	erode	accountant	a	02-JAN-00	8000
sumi	800	erode	accountant	h	02-MAR-00	17000
ravi	900	namakkal	accountant	k	02-FEB-00	6000

10 rows selected.

**A) Display the name of the employee whose salary is greater than Rs.10, 000.**

SQL> select employee \_name from employee where salary>10000;

EMPLOYEE\_NAME

-----

kumar

sumi

**B) Display the details of employees in ascending order according to employee code.**

SQL> select \* from employee ORDER BY employee\_code;

EMPL\_NA EMP\_CO ADDRESS DESIGNA G DATE\_OF\_J SALARY

-----

raj	200	karur	accountant	c	02-FEB-00	5000
kumar	300	tirupur	manager	d	01-MAR-00	12000
kaml	400	chennai	president	e	02-MAY-00	9000
kannan	500	madurai	manager	g	02-FEB-00	5000
tamil	600	madurai	manager	r	02-MAR-00	7000
suba	800	erode	accountant	a	02-JAN-00	8000
sumi	800	erode	accountant	h	02-MAR-00	17000
ravi	900	namakkal	accountant	k	02-FEB-00	6000
hari	1000	salem	manager	j	01-MAY-00	8000
mani	2000	coimbatore	chairman	a	01-JAN-00	10000

10 rows selected.

**C) Display the total salary of the employees whose grade is “A”.**

SQL> select sum(salary) from employee where grade='a';

SUM(SALARY)

-----

18000

**D) Display the details of the employee earning the highest salary.**

SQL> select \* from employee where salary=(select max(salary) from employee);

EMPL_NA	EMP_CO	ADDRESS	DESIGNA	G	DATE_OF_J	SALARY
---------	--------	---------	---------	---	-----------	--------

-----

sumi	800	erode	accountant	h	02-MAR-00	17000
------	-----	-------	------------	---	-----------	-------

**E) Display the names of the employees who earn more than “Ravi”.**

SQL> select employee\_name from employee where salary>(select salary from employee where employee\_name='ravi');

EMPLOYEE_NAME
---------------

-----

hari

mani

kumar

kaml

tamil

suba

sumi

7 rows selected.

**RESULT**

EX NO: 3	<b><u>STUDENT DETAILS</u></b>
DATE	

**AIM:**

To create a table named “students” with the fields name, gender, roll no, department, address and percentage.

**ALGORITHM:**

**Step 1:** start the program.

**Step 2:** create a student table using the create table query.

**Step 3:** insert the records into the student table using insert query.

**Step 4:** calculate the average percentage of the students by using avg ( ) function query.

**Step 5:** display the names of students whose percentage is greater than 80.

**Step 6:** by using select query and max ( ) function display the details of who has the second highest percentage.

**Step 7:** display the details of the students whose percentage is between 50 and to use a select query.

**Step 8:** to display the details of the student’s percentage greater than the roll no= 12CA01 use select query with the sub queries.

**Step 9:** step the process.

## **SQL QUERIES AND OUTPUT**

```
SQL> create table students(student_name char(10),gender char(6),roll_no varchar2(8),department_name char(10),address char(10),percentage number(4,2));
```

Table created.

```
SQL> insert into students values('arun','female','12CA12','commerce','ooty',90.0);
```

1 row created.

```
SQL> insert into students values('priya','female','12CA08','commerce','trichy',80.0);
```

1 row created.

```
SQL> desc students;
```

Name	Null?	Type
-----		
STUDENT_NAME		CHAR(10)
GENDER		CHAR(6)
ROLL_NO		VARCHAR2(8)
DEPARTMENT_NAME		CHAR(10)
ADDRESS		CHAR(10)
PERCENTAGE		NUMBER(4,2)

```
SQL> select * from students;
```

STUDENT_NA	GENDER	ROLL_NO	DEPARTMENT	ADDRESS	PERCENTAGE
-----	-----	-----	-----	-----	-----
a run	female	12CA12	commerce	ooty	90
priya	female	12CA08	commerce	trichy	80
kumar	male	12CA01	commerce	tirpur	75

ramya	female	12CA02	commerce	covai	80
anitha	female	12CA03	commerce	tripur	70
latha	female	12CA04	commerce	tripur	60
murugan	male	12CA05	commerce	trichy	40
raja	male	12CA06	commerce	covai	30
ram	male	12CA07	commerce	covai	20
Vicky	male	12CA013	commerce	annur	10
Vasanth	male	12CA023	commerce	chennai	70

11 rows selected.

**A) Calculate the average percentage of students.**

SQL> select avg(percentage)from students;

AVG(PERCENTAGE)

-----

56.8181818

**B) Display the names of the students whose percentage is greater than 80.**

SQL> select student\_name from students where percentage>80;

STUDENT\_NAME

-----

arun

**C) Display the details of the students who got the highest percentage.**

SQL> select \* from students where percentage=(select max(percentage)from students);



STUDENT_NA	GENDER	ROLL_NO	DEPARTMENT	ADDRESS	PERCENTAGE
-----	-----	-----	-----	-----	-----
Arun	female	12CA12	commerce	ooty	90

**D) Display the names of the students whose percentage is between 50 and 70.**

SQL> select \* from students where percentage between 50 and 70;

STUDENT_NA	GENDER	ROLL_NO	DEPARTMENT	ADDRESS	PERCENTAGE
-----	-----	-----	-----	-----	-----
Anitha	female	12CA03	commerce	tripur	70
Latha	female	12CA04	commerce	tripur	60
Vasanth	male	12CA023	commerce	chennai	70

**E) Display the details of the students whose percentage is greater than the percentage of the roll no=12CA01.**

SQL> select \* from students where percentage>(SELECT percentage from students whererooll\_no='12CA01');

STUDENT_NA	GENDER	ROLL_NO	DEPARTMENT	ADDRESS	PERCENTAGE
-----	-----	-----	-----	-----	-----
Arun	female	12CA12	commerce	ooty	90
Priya	female	12CA08	commerce	trichy	80
Ramya	female	12CA02	commerce	covai	80

**RESULT:**

EX NO: 4	<b>PRODUCT DETAILS</b>
DATE	

**AIM:**

To create a table with the name 'Product' with various field such as product number, name, unit of measure, quantity and total amount.

**ALGORITHM:**

**Step 1:** Start the program.

**Step 2:** Create a product table using the create table query.

**Step 3:** Insert the records into the product table using insert query.

**Step 4:** By using the update statement calculate the total amount.

**Step 5:** Display the records of the unit of measure ="kg" in the table by using a select query.

**Step 6:** By using select query displays the records whose quantity is greater than 10 and less than or equal to 20.

**Step 7:** Using sum operation calculate the total amount.

**Step 8:** Calculate the number of records whose unit price is greater than 10 by using count operation.

**Step 9:** Stop the process.

## SQL Queries and Output

```
SQL> create table product(product_no number(6),product_name char(10),unit_of_measure char(10),quantity number(6,4),total_amount number(8,4),unit_price number(6));
```

Table created.

```
SQL> insert into product values(111,'boost','kg',9.0,null,10);
```

1 row created.

```
SQL> insert into product values(222,'milk','l',2.0,null,15);
```

1 row created.

```
SQL> select * from product;
```

PRODUCT_NO	PRODUCT_NAME	UNIT_OF_MEASURE	QUANTITY	TOTAL_AMOUNT	UNIT_PRICE
-----	-----	-----	-----	-----	-----
111	boost	kg	9		10
222	milk	l	2		15
333	coffee	m	3		20
444	tea	m	6		25
555	cloth	m	5		30
666	cloth	m	4		35
777	water	l	3		40
888	juice	l	11		45
999	coffee	l	20		50
1000	biscuit	u	2		70

10 rows selected.

**A) using update statements calculate the total amount and then select the record.**

SQL> update product set total\_amount=(quantity\*unit\_price);

10 rows updated.

SQL> select \* from product;

PRODUCT_NO	PRODUCT_NA	UNIT_OF_ME	QUANTITY	TOTAL_AMOUNT	UNIT_PRICE
-----	-----	-----	-----	-----	-----
111	boost	kg	9	90	10
222	milk	l	2	30	15
333	coffee	m	3	60	20
444	tea	m	6	150	25
555	cloth	m	5	150	30
666	cloth	m	4	140	35
777	water	l	3	120	40
888	juice	l	11	495	45
999	coffee	l	20	1000	50
1000	biscut	u	2	140	70

10 rows selected.

**B) Select the records whose unit of measure is “kg”.**

SQL> select \* from product where unit\_of\_measure='kg';

PRODUCT_NO	PRODUCT_NA	UNIT_OF_ME	QUANTITY	TOTAL_AMOUNT	UNIT_PRICE
-----	-----	-----	-----	-----	-----
111	boost	kg	9	90	10

**C) Select the records whose quantity is greater than 10 and less than or equal to 20.**

SQL> select \* from product where quantity>=10 and quantity<=20;

PRODUCT_NO	PRODUCT_NA	UNIT_OF_ME	QUANTITY	TOTAL_AMOUNT	UNIT_PRICE
------------	------------	------------	----------	--------------	------------

888	juice	1	11	495	45
999	coffee	1	20	1000	50

**D) Calculate the entire total amount by using sum operation.**

SQL> select sum(total\_amount) from product;

SUM(TOTAL\_AMOUNT)

-----  
2375

**E) Calculate the number of records whose unit price is greater than 50 with count operation.**

SQL> select count(\*) from product where unit\_price>10;

COUNT(\*)

-----  
9

**RESULT**

EX NO: 5	<b>PAYROLL DETAILS</b>
DATE	

**AIM:**

To create a table with the name “payroll” with various fields such as employee number, name, department, basic pay, hra, da, pf and net pay.

**ALGORITHM:**

**Step 1:** Start the program.

**Step 2:** Create a payroll table using the create table query.

**Step 3:** Insert the records into the payroll table using insert query.

**Step 4:** By using update statement calculate the net pay.

**Step 5:** Display the records in ascending order by using order by query.

**Step 6:** By using select query displays the records whose department is sales.

**Step 7:** Display the details of the employees based on the condition  $HRA \geq 1000$  AND  $PA \leq 900$ .

**Step 8:** Display the records in descending order.

**Step 9:** Stop the process.

## **SQL Queries and Output**

```
SQL> create table payroll(employee_no number(4),employee_name char(5),depratmen  
t char(8),basic_pay number(6,2),hra number(4,2),da number(4,2),pf number(4,2),ne  
t_pay number(8,2));
```

Table created.

```
SQL> insert into payroll values(111,'niki','CS',7000,500,100,200, 0);
```

1 row created.

```
SQL> insert into payroll values(222,'sowmi','sales',10000,1000,200,420,0);
```

1 row created.

```
SQL>desc payroll;
```

Name	Null?	Type
-----		
EMPLOYEE_NO		NUMBER(8)
EMPLOYEE_NAME		VARCHAR2(10)
DEPRATMENT		VARCHAR2(10)
BASIC_PAY		NUMBER(8,2)
HRA		NUMBER(6,2)
DA		NUMBER(6,2)
PF		NUMBER(6,2)
NET_PAY		NUMBER(8,2)

SQL> select \* from payroll;

EMPLOYEE_NO	EMPLO	DEPRATME	BASIC_PAY	HRA	DA	PF	NET_PAY
-----	-----	-----	-----	-----	-----	-----	-----
111	niki	CS	7000	500	100	200	0
222	sowmi	sales	10000	1000	200	420	0
333	krish	production	15000	1500	500	500	0
444	guna	sales	10000	1000	350	200	0
555	saran	Accounts	7000	500	100	200	0
666	gokul	purchase	12000	1200	800	1000	0
777	shruthi	production	10000	2000	500	450	0
888	karthik	sales	8000	300	100	100	0
999	sivam	design	20000	2000	600	400	0
1000	hari	purchase	7000	500	100	200	0

10 rows selected.

**A) Update the records to calculate the net pay.**

SQL> update payroll set net\_pay=(basic\_pay+hra+da)-pf;

10 rows updated.

SQL> select \* from payroll;

EMPLOYEE_NO	EMPLO	DEPRATME	BASIC_PAY	HRA	DA	PF	NET_PAY
-----	-----	-----	-----	-----	-----	-----	-----
111	niki	CS	7000	500	100	200	7400
222	sowmi	sales	10000	1000	200	420	10780
333	krish	production	15000	1500	500	500	16500
444	guna	sales	10000	1000	350	200	11150



555	saran	Accounts	7000	500	100	200	4700
666	gokul	purchase	12000	1200	800	1000	13000
777	shruthi	production	10000	2000	500	450	12050
888	karthik	sales	8000	300	100	100	8300
999	sivam	design	20000	2000	600	400	22200
1000	hari	purchase	7000	500	100	200	7400

10 rows selected.

**b) Arrange the records of the employees in ascending order of their net pay.**

SQL> select \* from payroll order by net\_pay;

EMPLOYEE_NO	EMPLO	DEPRATME	BASIC_PAY	HRA	DA	PF	NET_PAY
-----	-----	-----	-----	-----	-----	-----	-----
----							
111	niki	CS	7000	500	100	200	7400
555	saran	Accounts	7000	500	100	200	7400
1000	hari	purchase	7000	500	100	200	7400
888	karthik	sales	8000	300	100	100	8300
222	sowmi	sales	10000	1000	200	420	10780
444	guna	sales	10000	1000	350	200	11150
777	shruthi	production	10000	2000	500	450	12050
666	gokul	purchase	12000	1200	800	1000	13000
333	krish	production	15000	1500	500	500	16500
999	sivam	design	20000	2000	600	400	22200

10 rows selected.

**C) Display the details of the employees whose department is “sales”.**

SQL> select \* from payroll1 where department='sales';

EMPLOYEE_NO	EMPLO	DEPRATME	BASIC_PAY	HRA	DA	PF	NET_PAY
-----	-----	-----	-----	-----	-----	-----	-----
222	sowmi	sales	10000	1000	200	420	10780
444	guna	sales	10000	1000	350	200	11150
888	karthik	sales	8000	300	100	100	8300

3 rows selected.

**d) Select the details of employees whose HRA>=1000 and DA<=900.**

SQL> select \* from payroll where hra>=1000 and da<=900;

EMPLOYEE_NO	EMPLO	DEPRATME	BASIC_PAY	HRA	DA	PF	NET_PAY
-----	-----	-----	-----	-----	-----	-----	-----
222	sowmi	sales	10000	1000	200	420	10780
333	krish	production	15000	1500	500	500	16500
444	guna	sales	10000	1000	350	200	11150
666	gokul	purchase	12000	1200	800	1000	13000
777	shruthi	production	10000	2000	500	450	12050
999	sivam	design	20000	2000	600	400	22200

6 rows selected.

e) Select the records in descending order.

SQL> select \* from payroll order by employee\_nodesc;

EMPLOYEE_NO	EMPLO	DEPRATME	BASIC_PAY	HRA	DA	PF	NET_PAY
-----	-----	-----	-----	-----	---	----	----
111	niki	CS	7000	500	100	200	7400
222	sowmi	sales	10000	1000	200	420	10780
333	krish	production	15000	1500	500	500	16500
444	guna	sales	10000	1000	350	200	11150
555	saran	Accounts	7000	500	100	200	4700
666	gokul	purchase	12000	1200	800	1000	13000
777	shruthi	production	10000	2000	500	450	12050
888	karthik	sales	8000	300	100	100	8300
999	sivam	design	20000	2000	600	400	22200
1000	hari	purchase	7000	500	100	200	7400

10 rows selected.

**RESULT**

EX NO: 6	<b>BOOK DETAILS</b>
DATE	

**AIM:**

To create a table with the name “Book” with the field publisher code, name, city, state, title of the book, book code and book price.

**ALGORITHM:**

**Step 1:** Start the program.

**Step 2:** Create a book table using the create table query.

**Step 3:** Insert the records into the book table using insert query.

**Step 4:** Display the table structure.

**Step 5:** Display the records with title as DBMS.

**Step 6:** By using select query displays the details of the book with the price above 300.

**Step 7:** Display the details of the book with the publisher name “Kalyani”.

**Step 8:** Calculate the number of books the publisher starts with “Sultan Chand” using count operation.

**Step 9:** Stop the process.

## **SQL Queries and Output**

```
SQL> create table book(publisher_code varchar2(5),publisher_name varchar2(10),publ  
isher_cityvarchar2(12),publisher_state varchar2(10),title of_book varchar2(15),boo  
k_codevarchar2(5),book_price varchar2(5));
```

Table created.

### **A) Insert the records into the table .**

```
SQL> insert into book values(111,'kalyani','delhi','delhi','DBMS',111,350);
```

1 row created.

```
SQL> insert into book values(222,'sulthan','kerala','kerala','c',222,450);
```

1 row created.

### **B) Describe the structure of the table.**

```
SQL>desc book;
```

Name	Null?	Type
-----		
PUBLISHER_CODE		VARCHAR2(5)
PUBLISHER_NAME		VARCHAR2(10)
PUBLISHER_CITY		VARCHAR2(12)
PUBLISHER_STATE		VARCHAR2(10)
TITLE_OF_BOOK		VARCHAR2(15)
BOOK_CODE		VARCHAR2(5)
BOOK_PRICE		VARCHAR2(5)

**C) Show the details of the book with the title 'DBMS'.**

SQL> select \* FROM book where title\_of\_book='DBMS';

PUBLI-CO	PUBLI_NA	PUBLISH_CI	PUBLISH_ST	TITLE_OF_BOOK	BOOK_C	BOOK_PR
----	-----	-----	-----	-----	----	----
111	kalyani	delhi	delhi	DBMS	111	350
555	kanimozhi	kerala	kerala	DBMS	555	400

**D) Show the details of the book with price >300.**

SQL> select \* from book where book\_price>300;

PUB_CO	PUBLI_NA	PUBLISH_CI	PUBLISHER_ST	TITLE_OF_BOOK	BOOK_CO	BOOK_PR
----	-----	-----	-----	-----	----	----
111	kalyani	delhi	delhi	DBMS	111	350
222	sulthan	kerala	kerala	c	222	450
333	kani	tamilnadu	tamilnadu	c	333	550
555	kanimozhi	kerala	kerala	DBMS	555	400
666	sindhu	delhi	delhi	c	666	600
777	keerthana	delhi	delhi	c	777	700
888	saranya	delhi	delhi	c	888	800
999	sandhya	tamilnadu	tamilnadu	c	899	900

8 rows selected.

**E) Show the details of the book with publisher name 'Kalyani'.**

SQL> select \* from book where publisher\_name='kalyani';

PUB_CO	PUBLI_NA	PUBLISH_CI	PUBLISHER_ST	TITLE_OF_BOOK	BOOK_CO	BOOK_PR
--------	----------	------------	--------------	---------------	---------	---------

111	kalyani	delhi	delhi	DBMS	111	350
-----	---------	-------	-------	------	-----	-----

**F) Select the book code, book title, publisher city is 'Delhi'.**

SQL> select book\_code,titleof\_book FROM book where publisher\_city='delhi';

BOOK_CODE	TITLE_OF_BOOK
-----------	---------------

-----

111	DBMS
-----	------

666	c
-----	---

777	c
-----	---

888	c
-----	---

**G) Select the book code, book title and sort by book price.**

selectbook\_code,titleof\_book FROM ORDER BY book\_price;

BOOK_CODE	TITLE_OF_BOOK
-----------	---------------

-----

111	DBMS
-----	------

333	c
-----	---

555	DBMS
-----	------

666	c
-----	---

777	c
-----	---

888	c
-----	---

999	c
-----	---

**H) Count the number of books of publisher starts with “sultan chand”.**

SQL> select count(\*) from book WHERE publisher\_name='sulthan';

COUNT(\*)

-----

1

**I) find the name of the publisher starting with “S”.**

SQL> select publisher\_name from book where publisher\_name like 'S%';

PUBLISHER\_NAME

-----

sulthan

sanjay

sindhu

saranya

sandhya

**RESULT:**



EX NO: 7	<b>BANK DETAILS</b>
DATE	

**AIM:**

To create a table with the name “loan” with various fields such as account, branch name, customer name, balance amount, loan number and loan amount=t.

**ALGORITHM:**

**Step 1:** Start the program.

**Step 2:** Create a loan table using the create table query.

**Step 3:** Insert the records into the loan table using insert query.

**Step 4:** Display the records of the table.

**Step 5:** Display the number of loans with amounts between 1000 and 5000.

**Step 6:** Display customer’s names in alphabetical order that have load at Coimbatore branch.

**Step 7:** Using avg operation calculate the amount balance at Coimbatore branch.

**Step 8:** Stop the process.

## **SQL Queries and Output**

```
SQL> create table loan(account varchar2(6),branch_name varchar2(12),customer_name  
Varchar2(7),balance_amount varcha2r(8),loan_number varchar2(7),loan_amount varchar2  
(6));
```

Table created.

### **a) Insert the records into the table.**

```
SQL> insert into loan values(111,'coimbatore','arun',1000,111,15000);
```

1 row created.

```
SQL> insert into loan values(222,'coimbatore','ram',1000,222,25000);
```

1 row created.

### **b) Describe the structure of the table.**

```
SQL>desc loan;
```

Name	Null?	Type
-----		
ACCOUNT		VARCHAR2(6)
BRANCH_NAME		VARCHAR2(12)
CUSTOMER_NAME		VARCHAR2(7)
BALANCE_AMOUNT		VARCHAR2(8)
LOAN_NUMBER		VARCHAR2(7)
LOAN_AMOUNT		VARCHAR2(6)

**c) Display the records of Deposit and Loan.**

SQL> select \* from loan;

ACCOUN	BRANCH_NAME	CUSTOMER	BALANCE_AMT	LOAN_NUM	LOAN_AMO
-----	-----	-----	-----	-----	-----
111	coimbatore	arun	1000	111	15000
222	coimbatore	ram	1000	222	25000
333	coimbatore	raj	1000	333	8000
444	erode	hari	1000	444	7000
555	erode	hari	1000	555	6000
666	erode	kumar	1000	444	5000
777	thirupur	karthi	1000	333	4000
888	thirupur	kishor	1000	666	3000
999	thirupur	kavi	1000	777	2000
1000	thirupur	ramiya	1000	888	1000

10 rows selected.

**d) Find the number of loans with amounts between 10000 and 50000.**

SQL> select count(\*)from loan1 where loan\_amount between 10000 and 50000;

COUNT(\*)

-----

**e) List in the alphabetical order the names of all customers who have a loan at the Coimbatore branch.**

```
SQL> select customer_name from loan where branch_name='coimbatore' order by  
customer_name;
```

CUSTOMER

-----

arun

raj

ram

**f) Find the average account balance at the Coimbatore branch.**

```
SQL> select avg(balance_amount) from loan where branch_name='coimbatore';
```

AVG(BALANCE\_AMOUNT)

-----

1000

**g) Update deposits to add interest at 5% to the balance.**

```
SQL> update loan1 set balance_amount=balance_amount+(balance_amount*.5);
```

10 rows updated.

```
SQL> select * from loan1;
```

ACCOUNT	BRANCH_NAME	CUSTOMER	BALANCE_AMO	LOAN_NUM	LOAN_AMO
-----	-----	-----	-----	-----	-----
111	coimbatore	arun	2250	111	15000
222	coimbatore	ram	2250	222	25000
333	coimbatore	raj	2250	333	8000
444	erode	hari	2250	444	7000

555	erode	hari	2250	555	6000
666	erode	kumar	2250	444	5000
777	thirupur	karthi	2250	333	4000
888	thirupur	kishor	2250	666	3000
999	thirupur	kavi	2250	777	2000
1000	thirupur	ramiya	2250	888	1000

10 rows selected.

**h) Arrange the records in descending order of the loan amount.**

SQL> select \* from loan1 order by loan\_amount desc;

ACCOUN	BRANCH_NAME	CUSTOME	BALANCE_	LOAN_NU	LOAN_A
-----	-----	-----	-----	-----	-----
333	coimbatore	raj	1000	333	8000
444	erode	hari	1000	444	7000
555	erode	hari	1000	555	6000
666	erode	kumar	1000	444	5000
777	thirupur	karthi	1000	333	4000
888	thirupur	kishor	1000	666	3000
222	coimbatore	ram	1000	222	25000
999	thirupur	kavi	1000	777	2000
111	coimbatore	arun	1000	111	15000
1000	thirupur	ramiya	1000	888	1000

**i) Find the total amount of deposit in 'Erode' branch.**

SQL> select sum(balance\_amount) from loan1 where branch\_name='erode';

SUM(BALANCE\_AMOUNT)

-----

4500

**RESULT:**

C++

EX NO: 1

DATE

## STRAIGHT LINE METHOD

### AIM:

To create a C++ Program to calculate depreciation under straight line method using class by defining member function outside the class

### ALGORITHM:

1. Start the Program
2. Include the Header files.
3. Define class and declare the variables and method in the required access specified.
4. Define the member function outside the class.
5. Get the required values and use the formula  
Straight-line Depreciation Expense =  $\frac{\text{cost} - \text{Salvage value}}{\text{Useful life of Asset}}$

6. Create an object for the class in the main program to call the member functions of the class.
7. Run the program and display the results.
8. Stop the program.



## **PROGRAM:**

```
#include<iostream.h>
#include<conio.h>
class dep
{
private:
    float cost,scrap,life_time,depexp;
public:
    void getdata();
    void calculation();
};
void dep::getdata()
{
    cout<<"/n/n enter the cost of asset:";
    cin>>cost;
    cout<<"enter the scrap value";
    cin>>scrap;
    cout<<"enter the usefull life of asset";
    cin>>life_time;
}
void dep::calculation()
{
    cout<<"/n calculation the annual depreciation expense under straight line method";
    cout<<"/n*****
*****/n";
    depexp=(cost-scrap)/life_time;
    cout<<"cost of asset"<<cost<<"/n";
```

```
cout<<"scrap value"<<scrap<<"/n";
cout<<"useful life time of asset"<<life_time;
cout<<"/n/n annual deprrciation expense using straight line method:"<<depexp;
};
void main()
{
clrscr();
cout<<"/n calculation of depreciation using straight line method";
cout<<"/n*****/n";
dep dp;
dp.getdata();
dp.calculation();
getch();
}
```

**OUTPUT:**

calculation of depreciation using straight line method

enter the cost of asset:25000

enter the scrap value1200

enter the usefull life of asset4

calculation the annual depreciation expense under straight line method

.....  
cost of asset25000

scrap value1200

useful life time of asset 4

annual deprrciation expense using straight line method:5950

**RESULT:**

EX NO: 2	<b>DIMINISHING BALANCE METHOD</b>
DATE	

**AIM:**

To create a C++ Program to calculate depreciation under Diminishing Balance Method using class by defining member function inside the class.

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define class and declare the variables and methods in the required access specifier.
4. Define the member function inside the class.
5. Get the required values and declare in the functions.
6. Use the formula  $(\text{Amount} * \text{Rate} / 100)$  to calculate the Depreciation and formula  $(\text{Amount} - \text{Depreciation})$  to calculate the Balance.
7. Create an object for the class in the main program to call the member functions of the class.
8. Run the program and display the results.
9. Stop the program.

## **PROGRAM:**

```
#include<iostream.h>

#include<conio.h>

class dep
{
private:
float amount,rate,fd,sd,td,bal1,bal2,bal3;
public:
void getdata()
{
cout<<"\n\n enter the amount:";
cin>>amount;
cout<<"\n\n enter the rate:";
cin>>rate;
}
void calculation()
{
cout<<"\n\n statement of result";
cout<<"\n\n*****";
fd=(amount*rate/100);
bal1=(amount-fd);
cout<<"\n\n\n first year depreciation:"<<fd;
```

```
cout<<"\n\n first year balance:"<<bal1;

sd=(bal1*rate/100);

bal2=(bal1-sd);

cout<<"\n\n second year depreciation:"<<sd;

cout<<"\n\n second year balance:"<<bal2;

td=(bal2*rate/100);

bal3=(bal2-td);

cout<<"\n\n\n third year depreciation:"<<td;

cout<<"\n\n third year balance:"<<bal3;

}

};

void main()

{

cout<<"\n\n\t diminishing balance method";

cout<<"\n\n\t*****\n";

dep d;

d.getdata();

d.calculation();

getch();

}
```

## **OUTPUT**

\*\*\*\*\*

enter the amount:150000

enter the rate:10

### Statement of result

\*\*\*\*\*

first year depreciation:15000n

first year balance:135000

second year depreciation:13500

second year balance:121500

third year depreciation:12150

third year balance:109350

## **RESULT**

EX NO: 3	<b>ECONOMIC ORDER QUANTITY</b>
DATE	

**AIM:**

To create a C++ Program to calculate Economic Order Quantity using class by defining nesting of member functions.

**ALGORTITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define class and declare the variables and methods in the required access specified.
4. Define the member function inside the class.
5. Calculate the EOQ using the formula.
6. Create an object for the class in the main program to call the member functions of the class.
7. Run the program and display the results.
8. Stop the program



## **PROGRAM :**

```
#include<iostream.h>

#include<conio.h>

#include<math.h>

class eoq
{
private:
float a,b,c,q1,q2,quantity,rate;
public:
void getdata()
{
cout<<"\n\n enter the annual consumption:";
cin>>a;
cout<<"\n\n enter the order per unit:";
cin>>b;
cout<<"\n\n enter the cost perunit:";
cin>>c;
cout<<"\n\n enter the carrying cost:";
cin>>rate;
calculation();
}
void calculation()
{
```

```
cout<<"\n\n STATEMENT OF RESULT\n";
```

```
cout<<"\n\n*****\n\n";
```

```
q1=2*a*b;
```

```
q2=c*rate/100;
```

```
quantity=sqrt(q1/q2);
```

```
cout<<"EOQ:"<<quantity;
```

```
}
```

```
};
```

```
void main()
```

```
{
```

```
clrscr();
```

```
cout<<"CALCULATION OF EOQ\n";
```

```
cout<<"*****\n\n";
```

```
eoq e;
```

```
e.getdata();
```

```
getch();
```

```
};
```

**OUTPUT:**

**CALCULATION OF EOQ**

\*\*\*\*\*

enter the annual consumption:4000

enter the order per unit:1

enter the cost perunit:2

enter the carrying cost:10

**STATEMENT OF RESULT**

\*\*\*\*\*

EOQ:200

**RESULT:**

EX NO: 4	<b>PAYROLL MANAGEMENT</b>
DATE	

**AIM:**

To print the Employees' payroll statement (using control structures).

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define class and declare the variables and methods in the required access specifier.
4. Define the member function using nesting of member function.
5. Use control statements to check the age.
6. In the main function use do while statement to repeat the program execution.
7. Create an object name to call the function in the main function.
8. Run the program and display the results.
9. Stop the program

## **PROGRAM:**

```
#include<iostream.h>

#include<conio.h>

class employee
{
private:
char name[15],dept[10];
int id,age,nowd,pds;
float netsalary,allowance,basicsalary;
float da,hra,esi,pf,ta;
public:
void getdata();
void calculate();
void showresult();
};

void employee::getdata()
{
cout<<"\n employee payroll system";
cout<<"\n*****";
cout<<"\n enter the name:";
cin>>name;
cout<<"\n enter the department:";
cin>>dept;
```

```
cout<<"\n enter the employee id:";
cin>>id;
cout<<"\n enter the employee age:";
cin>>age;
if(age<18)
{
cout<<"\n valid age";
getdata();
}
else
{
cout<<"\n enter the number of working days;";
cin>>nowd;
cout<<"\n enter the per day salary:";
cin>>pds;
}
}
void employee::calculate()
{
int i,n,p,u,t;
basicsalary=nowd*pds;
cout<<"\n enter the da:";
cin>>i;
```

```
da=basicsalary*i/100;
cout<<" enter the hra:";
cin>>n;
hra=basicsalary*n/100;
cout<<"\n enter the esi:";
cin>>p;
esi=basicsalary*p/100;
cout<<"\n enter the ta:";
cin>>u;
ta=basicsalary*u/100;
cout<<"\n enter the pf:";
cin>>t;
pf=basicsalary*t/100;
allowance=da+hra+esi+ta+pf;
netsalary=basicsalary+allowance;
}

void employee::showresult()
{
cout<<"\n salary details:";
cout<<"\n*****";
cout<<"\n name:"<<name;
cout<<"\n id:"<<id;
cout<<"\n basicsalary:"<<basicsalary;
```

```
cout<<"\n da:"<<da;

cout<<"\n hra:"<<hra;

cout<<"\n esi:"<<esi;

cout<<"\n ta:"<<ta;

cout<<"\n pf:"<<pf;

cout<<"\n\n tatal allowance:"<<allowance;

cout<<"\n\n*****";

cout<<"\n\n netsalary:"<<netsalary;

cout<<"\n\n*****";

}

void main()

{ char ch;

do

{ clrscr();

employee e;

e.getdata();

e.calculate();

e.showresult();

getch();

cout<<"\n do you want to continue?(y/n):";

cin>>ch;

}while(ch==1);}
```



## **OUTPUT:**

### **EMPLOYEE PAYROLL SYSTEM**

\*\*\*\*\*

enter the employee name:Rajesh

enter the department:Production

enter the employee id:2314

enter the employee age:30

enter the number of working days:25

enter the perday salary:25500

enter the DA:10

enter the HRA:10

enter the esi:2

enter the TA:5

enter the pf:5

### **SALARY DETAILS:**

\*\*\*\*\*

name:Rajesh

id:7168

basicsalary:-17860

DA:-1786

HRA:-1786

ESI:-357.200012

TA:-893 PF:-893

Total Allowance:-3929.199951

\*\*\*\*\*

Netsalary:-21789.199219

\*\*\*\*\*

do you want to continue?(o/r:n

**RESULT:**

EX NO: 5	<b>SIMPLE AND COMPOUND INTEREST</b>
DATE	

**AIM:**

To calculate simple interest and compound interest(using nested class).

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define class with inheritance class with the variables and member functions.
4. The member function is declared inside the nested class.
5. To calculate simple interest use the formula  $(P*N*R)/100$ . And to calculate the compound interest use the formula  $(P*(1+R/100)^N)$
6. Create an object name for the both the class using scope resolution operator to call the function in the main program.
7. Execute the program and display the result.
8. Stop the process.

**PROGRAM:**

```
#include<iostream.h>

#include<conio.h>

#include<math.h>

class sinterest

{

public:

class cinterest

{

private:

double p,si,ci;

float n,r;

public:

void getdata()

{

cout<<"enter the principle amount:";

cin>>p;

cout<<"enter the number of years:";

cin>>n;

cout<<"enter the rate of interest:";

cin>>r;

}

void calculation()
```

```

{
cout<<"RESULT:";

cout<<"*****:";

si=(p*n*r)/100;

cout<<"simple interest:"<<si;

ci=p*pow((1+(r/100)),n)-p;

cout<<"compound interest:"<<ci;

}  };

};

void main()

{

clrscr();

sinterest::cinterest obj;

cout<<"SIMPLE INTEREST AND COMPOUND INTEREST:";

cout<<"*****:";

obj.getdata();

obj.calculation();

getch();

}

```

## **OUTPUT**

### **SIMPLE INTEREST AND COMPOUND INTEREST:**

\*\*\*\*\*.

enter the principle amount:5000

enter the number of years:5

enter the rate of interest:1

### **RESULT:**

simple interest:250

.....  
compound interest:255.050251

## **RESULT:**

EX NO: 6	<b>NET INCOME OF A FAMILY</b>
DATE	

**AIM:**

To calculate simple interest and compound interest(using nested class).

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define various classes with the member functions.
4. Define the friend function in both the classes doctor and advocate.
5. Access the member function using the scope resolution operator.
6. Create an object for the both the classes and call the function using the object name in the main program.
7. Run the program and display the results.
8. Stop the program.

## **PROGRAM**

```
#include<iostream.h>

#include<conio.h>

class advocate;

class doctor
{
private:
double vf,cf,sm,de,eb,d;
public:
void getdata();
void calculate();
friend void incometax(doctor,advocate);
};

class advocate
{
private:
double sc,pf,le,oe,ss,a;
public:
void getdata();
void calculate();
friend void incometax(doctor,advocate);
};

void doctor::getdata()
```



```

{
cout<<"\n\n enter visiting fee:";
cin>>vf;
cout<<"\n\n enter consulting fee:";
cin>>cf;
cout<<"\n\n enter sales of medicine:";
cin>>sm;
cout<<"\n\n enter e_bill:";
cin>>eb;
cout<<"\n\n enter dispensary expenses:";
cin>>de;
}

void doctor::calculate()
{
d=vf+cf+sm-(de+eb);
cout<<"\n\n_____";
cout<<"\n\n doctor professional gain is:rs."<<d;
cout<<"\n\n_____";
}

void advocate::getdata()
{
cout<<"\n\n enter the special commission:";
cin>>sc;

```

```

cout<<"\n\n enter the practicing fee:";
cin>>pf;
cout<<"\n\n enter the legal expenses:";
cin>>le;
cout<<"\n\n enter the office expenses:";
cin>>oe;
cout<<"\n\n enter the staff salary:";
cin>>ss;
}
void advocate::calculate()
{
a=sc+pf+le-(oe+ss);
cout<<"\n_____ \n";
cout<<"\n advocate professional gain is:rs."<<a;
cout<<"\n_____ \n";
}
void incometax(doctor c,advocate v)
{
double s=c.d+
v.a;
cout<<"\n_____ \n";
cout<<"\n net income:rs."<<s;
cout<<"\n_____ \n";

```

```
}  
  
void main()  
{  
clrscr();  
cout<<"NET INCOME OF A FAMILY \n";  
cout<<"*****\n\n";  
doctor r;  
r.getdata();  
r.calculate();  
advocate e;  
e.getdata();  
e.calculate();  
incometax(r,e);  
getch();  
}
```

## **OUTPUT**

enter the special commission:1500

enter the practicing fee:3000

enter the legal expenses:5000

enter the office expenses:1500

enter the staff salary:2000

---

advocate professional gain is: Rs.6000

---

---

net income: Rs.8000

---

## **RESULT:**

EX NO: 7	<b>LIBRARY BOOK DETAILS</b>
DATE	

AIM:

To create the C++ Program to print the book details of the library using array of objects.

ALGORITHM:

1. Start the Program.
2. Include the Header Files.
3. Define a class library with the member functions
4. Create array of objects in the main program.
5. Use for loop to call the array of objects.
6. Run the program and display the results.
7. Stop the program.

**PROGRAM:**

```
#include<iostream.h>

#include<conio.h>

class library
{
    int bookno,isbno;
    char bookname[10];
    char bookauthorname[20];
    public:
        void getdata();
        void display();
};

void library::getdata()
{
    cout<<"enter details of library book";
    cout<<"enter the bookno:";
    cin>>bookno;
    cout<<"enter the bookname:";
    cin>>bookname;
    cout<<"enter the book authorname:";
    cin>>bookauthorname;
    cout<<"enter the ISBN number:";
    cin>>isbno;
```

```

}

void library::display()
{
    cout<<"\n bookno:"<<bookno;
    cout<<"\n bookname:"<<bookname;
    cout<<"\n book authorname:"<<bookauthorname;
    cout<<"\n isb no:"<<isbno;
}

void main()
{clrscr();

    const int size=2;

    library book [size];

    cout<<"\n book list in librray\n";

    cout<<"\n*****\n";

    cout<<"\n total books:"<<size<<endl;

    for(int i=0;i< size;i++)
    { book[i].getdata();}

    cout<<"\n library book list";

    for(i=0;i< size;i++)
    { book[i].display();}

    getch();}

```

## **OUTPUT**

book list in library

\*\*\*\*\*

total books:2

enter details of library book enter the bookno:1211

enter the bookname:C++

enter the book authorname:Balagurusamy

enter the ISBN number:342311

enter details of library book enter the bookno:1212

enter the bookname:COBOL

enter the book authorname:Richard

enter the ISBN number:432421

library book list

bookno:1211

bookname:C++

book authorname:Balagurusamy

isb no:14631

bookno:1212

bookname:COBOL

book authorname:Richard

isb no:-26331

## **RESULT:**



EX NO: 8	COST SHEET
DATE	

AIM:

To prepare a cost sheet (using inheritance).

ALGORITHM:

1. Start the Program.
2. Include the Header Files.
3. Define a class member function and declare another member function in the inherited class.
4. Define the derived class sheet from the base class cost.
5. Create an object for the derived class sheet and call the function by using the object name in the main program.
6. Run the program and display the results.
7. Stop the program.

## **PROGRAM**

```
#include<iostream.h>

#include<conio.h>

class cost
{
public:
double de,ine,fe,se,oe,sales;

void getdata();

};

void cost::getdata()
{
clrscr();

cout<<"cost sheet:";

cout<<"*****";

cout<<"enter direct expenses:";

cin>>de;

cout<<"enter indirect expenses:";

cin>>ine;

cout<<"enter factory expenses:";

cin>>fe;

cout<<"enter selling expenses:";

cin>>se;

cout<<"enter office expenses:";
```

```

cin>>oe;

cout<<"enter sales:";

cin>>sales;

}

class sheet:public cost
{
public:

double primecost,workcost,costofproduction,costofsales,profit;

void display();

};

void sheet::display()
{

cout<<"\n\n\t calculation of cost sheet\n";

primecost=de+ine;

cout<<"primecost:"<<primecost;

workcost=primecost+fe;

cout<<"workcost:"<<workcost;

costofproduction=workcost+oe;

cout<<"costofproduction:"<<costofproduction;

costofsales=costofproduction +se;

cout<<"costofsales:"<<costofsales;

profit=sales-costofsales;

cout<<"profit:"<<profit;

```

```
}  
  
void main()  
{  
    clrscr();  
    sheet obj;  
    obj.getdata();  
    obj.display();  
    getch();  
}
```

## **OUTPUT**

cost sheet

\*\*\*\*\*

enter direct expenses:1500

enter indirect expenses:800

enter factory expenses:450

enter selling expenses:400

enter office expenses:600

enter sales:6000

Calculation of cost sheet

Primecost:2300

Workcost:2750

Cost of production:3350

Cost of sales:3750

Profit:2250

## **RESULT**

EX NO: 9	<b>MARGIN OF SAFETY</b>
DATE	

**AIM:**

To calculate margin of safety (using multilevel inheritance).

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define a class with a class with name as a and declare the variables.
4. Define derive class with name as b from the base class a and declare the member functions.
5. Define a one more derived class with name as c to inherit the class b and declare the member functions.
6. Thus the multilevel inheritance concept is implemented by deriving various classes one another.
7. Create an object for class c in the main program to call the member functions of class b .
8. Run the program to display the results.
9. Stop the process.

## **PROGRAM**

```
#include<iostream.h>

#include<conio.h>

class a
{
public:
float contribution,sales,vcost,profit,fcost,marginofsafety,pvratio;
};

class b:public a
{
public:
void getdata()
{
cout<<"\n\n\tMARGINOFSAFETY";
cout<<"\n\n\tenter the sales";
cin>>sales;
cout<<"\n\n\tenter the variablecost";
cin>>vcost;
cout<<"\n\n\tenter the fixedcost";
cin>>fcost;
} };

class c:public b
{
```

```
public:
void calculate()
{
contribution=(sales-vcost);
profit=contribution-fcost;
cout<<"\n\n\tRESULT";
cout<<"\n\n\tcontribution:"<<contribution;
cout<<"\n\n\tprofit:"<<profit;
pvratio=(contribution/sales)*100;
cout<<"\n\n\tpvratio:"<<pvratio;
marginofsafety=profit/pvratio;
cout<<"\n\n\tMARGINOFSAFETY:"<<marginofsafety;
} };
void main()
{
clrscr();
c d;
d.getdata();
d.calculate();
getch();
}
```



## **OUTPUT**

MARGINOFSAFETY

\*\*\*\*\*

enter the sales 10000

enter the variablecost1000

enter the fixedcost500

contribution:9000

profit:8500

pvratio:90

MARGINOFSAFETY:94.444443

## **RESULT**

EX NO: 10	<b>BANK TRANSACTION</b>
DATE	

**AIM:**

To create a C++ Program to print the Bank Transaction details using constructor and destructor.

**ALGORITHM**

1. Start the Program.
2. Include the Header Files.
3. Define a class with a name bank.
4. Define the member functions inside the class.
5. Create a constructor and destructor for the class bank.
6. Create an object name to call the member function of the class from the main program to get the value of customer details with transaction amount and to select the Transaction type.
7. Create the Switch case statement in the main program to enter the choice of operation in Banking.
8. Run the program and display the result.
9. Stop the process.

## **PROGRAM**

```
#include<iostream.h>

#include<conio.h>

class bank
{
private:
int accno,amount,wamt,damt;
char name[25],desi[25],cty[25];
public:
bank()
{
accno=1001;
}
~bank()
{
cout<<"\n\n****happy banking****:";
}
void getdata()
{
cout<<"\n\ncustomer number:"<<accno;
cout<<"\n\nenter the customer name:";
cin>>name;
cout<<"\n\nenter the designation:";
```

```
cin>>desi;

cout<<"\n\nenter the city:";

cin>>cty;

cout<<"\n\nenter the amount:";

cin>>amount;

}

void withdraw()

{

cout<<"\n\nwithdraw\n";

cout<<"\n\n-----\n";

cout<<"enter the amount:";

cin>>want;

amount=amount-wamt;

cout<<"\nbalance:"<<amount;

}

void deposit()

{

cout<<"\n\ndeposit\n";

cout<<"\n\n-----\n";

cout<<"enter the amount:";

cin>>damt;

amount=amount+damt;

cout<<"\nbalance:"<<amount;
```

```
}  
};  
  
void main()  
{  
    bank bk;  
    int ch;  
    clrscr();  
    cout<<"state bank of india\n";  
    cout<<"*****";  
    bk.getdata();  
    cout<<"\n\ntransaction";  
    cout<<"\n*****\n";  
    cout<<"1.deposit\n 2.withdraw\n\n";  
    cout<<"\n\nenter your choice:";  
    cin>>ch;  
    switch(ch)  
    {  
        case 1:  
            bk.deposit();break;  
        case 2:bk.withdraw();break;  
        default:cout<<"\n*****corong option*****\n";  
    }getch();}
```

## **OUTPUT**

enter the city:Coimbatore

enter the amount:10000

transaction

\*\*\*\*\*

1.deposit

2.withdraw

enter your choice:1

deposit

-----

enter the amount:5000

balance:15000

\*\*\*happy banking\*\*\*:

## **RESULT**

EX NO: 11	<b>WORKING CAPITAL</b>
DATE	

**AIM:**

To create a C++ Program to calculate increase or decrease in working capital using operator overloading.

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. Define two classes with name asset and liability.
4. Define the member functions and declare variables in both the class.
5. Define the operator overloading function using – symbol
6. Declare the operator overloading function in both classes.
7. Create the objects for both the classes to call the member functions to get the values for calculate the asset and liability value.
8. Call the operator function to calculate the value of working capital.
9. Run the program to display the results.
10. Stop the process.

## **PROGRAM**

```
#include<iostream.h>

#include<conio.h>

class liability;

class asset

{

private:

double cash,drs,stock,ast;

public:

void getdata()

{

cout<<"\current asset\n";

cout<<"-----\n";

cout<<"\ncash:";

cin>>cash;

cout<<"\ndebtors:";

cin>>drs;

cout<<"\nstock:";

cin>>stock;

ast=cash+drs+stock;

}

friend void operator-(asset a,liability l);

};
```



```

class liability
{
private:
double bp,tp,crs,lib;
public:
void getdata()
{
cout<<"\ncurrent liability\n";
cout<<"\n-----\n";
cout<<"\nbill payable:";
cin>>bp;
cout<<"\ntax payable:";
cin>>tp;
cout<<"\ncreditors:";
cin>>crs;
lib=bp+tp+crs;
}
friend void operator-(asset a,liability l)
};
void operator-(asset a,liability l)
{
double wc;
wc=a.ast-l.lib;

```

```
cout<<"\n\n-----\n:";

cout<<"\n WORKING CAPTIAL:"<<wc;

cout<<"\n\n-----\n:";

}

void main()

{

clrscr();

cout<<"WORKIMNG CAPTIAL:";

cout<<"*****\n\n:";

asset a;

a.getdata();

liability l;

l.getdata();

a-l;

getch();

}
```

## **OUTPUT**

### **WORKING CAPITAL**

\*\*\*\*\*

#### **CURRENT ASSEST**

---

cash:1000

debtors:200

stock:300

current liability

---

bill payable:100

tax payable:100

creditors:100

\*\*\*\*\*

**WORKING CAPTIAL : 1200**

\*\*\*\*\*

## **RESULT**

EX NO: 12	<b>STUDENTS MARK STATEMENT</b>
DATE	

**AIM:**

To create a C++ Program to display the students file and prepare the marks slip by accessing the file.

**ALGORITHM:**

1. Start the Program.
2. Include the Header Files.
3. In the main function, declare objects for the ofstream and ifstream.
4. Get the file name and open the file using the open() in ofstream .
5. Write the students details into the file and calculating the total and average.
6. Close the file.
7. Open the file ifstream to read and display the results.
8. A text file will be created as output.
9. Run the program to display the results.
10. Stop the process.

## **PROGRAM:**

```
#include<iostream.h>

#include<conio.h>

#include<fstream.h>

void main()

{

char rno[5],name[10],fname[25];

int m1,m2,m3,total;

float aver;

ofstream fout;

ifstream fin;

clrscr();

cout<<"STUDENT MARK DETAILS\n";

cout<<"*****\n";

cout<<"\n\nenter the file name with extension(.txt):";

cin>>fname;

fout.open(fname);

cout<<"\n\nenter the details";

cout<<"\n\nroll no:";

cin>>rno;

fout<<rno<<"";

cout<<"\n\nname:";

cin>>name;

fout<<name<<"";

cout<<"\n\nmark1L:";
```

```
cin>>m1;
fout<<m1<<"";
cout<<"\n\nmark2:";
cin>>m2;
fout<<m2<<"";
cout<<"\n\nmark3:";
cin>>m3;
fout<<m3<<"";
total=m1+m2+m3;
fout<<total<<"";
aver=total/3;
fout<<aver<<"";
fout.close();
cout<<"\n\nSTATEMENT OF MARKS\n";
cout<<"*****\n";
fin.open(fname);
cout<<"\n\nroll no:";
fin>>rno;
cout<<"\n\nname:";
fin>>name;
cout<<"\n\nname";
cout<<"\n\nmark1:";
fin>>m1;
cout<<m1;
cout<<"\n\nmark2:";
```

```
fin>>m2;
cout<<m2;
cout<<"\n\nmark3:";
fin>>m3;
cout<<m3;
cout<<"\n\ntotal:";
fin>>total;
cout<<total;
cout<<"\n\naverage:";
fin>>aver;
cout<<aver;
fin.close();
getch();
}
```

## **OUTPUT**

### **SUDENTS MARK DETAILS**

\*\*\*\*\*

Enter the file name with extension (.txt) : II B.Com PA.txt

Enter the details

Roll no: 100

Name : AAA

Mark1 : 90

Mark2: 89

Mark3: 77

### **STATEMENT OF MARKS**

\*\*\*\*\*

roll no: 100

name: AAA

mark1: 90

mark2: 89

mark3: 77

total: 256

average:- 85

## **RESULT**