**C PROGRAMMING**

**PROGRAM NO: 1**

**AIM: Simple program using formatted I/O.**

**ALGORITHM**

Step 1 : start

Step 2 : read values of a and b from user

Step 3 : print the values of a and b

Step 4 : read value of n from user

Step 5 : print the value of n, right justified in the field

of 6 columns

Step 6 : print the value of n, right justified in the field

of 2 columns

Step 7 : print the value of n, left justified in the field

of 6 columns

Step 8 : print the value of n, placing zeros before n in the field

of 6 columns

Step 9 : read values of x and y from user

Step 10 : read value of w from user

Step 11 : read value of p from user

Step 12 : print the values of x, is rounded to p decimal places

and right justified in the field of w columns

Step 13 : print the value of x in exponential form with

default precision 6

Step 14 : print the value of y, is rounded to 2 decimal places and

left justified in the field of 7 columns

Step 15 : print the value of y in exponential form , is rounded

to 2 decimal places and right justified in the field

of 10 columns

Step 16 : read value of str from user

Step 17 : print the value of str in the field of 20 columns

Step 18 : print the first 10 characters of the string str in the field

of 30 columns and right justified

Step 19 : print the first 5 characters of the string str

Step 20 : print the first 10 characters of the string str in the field

of 20 columns and left justified

Step 21 : print the value of str

Step 22 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int a,b,d,w,p;

float x;

double y;

char str[20];

printf("FORMATTED INPUT & OUTPUT\n");

printf("Enter three integer numbers\n");

scanf("%d%\*d%d",&a,&b);

printf("value in a is%d\n Value in b is %d\n",a,b);

printf("Enter a 4 digit number\n");

scanf("%4d",&d);

printf("%6d\n",d);

printf("%2d\n",d);

printf("%-6d\n",d);

printf("%06d\n",d);

printf("Enter two floating point number\n");

scanf("%f%f",&x,&y);

printf("Enter the field width\n");

scanf("%d",&w);

printf("Enter the precision\n");

scanf("%d",&p);

printf("%\*.\*f\n",w,p,x);

printf("%e\n",x);

printf("%-7.2f\n",y);

printf("%10.2e\n",y);

printf("Enter a string\n");

scanf("%s",str);

printf("%20s\n",str);

printf("%30.10s\n",str);

printf("%.5s\n",str);

printf("%-20s\n",str);

printf("%5s\n",str);

}

**OUTPUT:**

FORMATTED INPUT & OUTPUT

Enter three integer numbers

1

2

3

value in a is1

Value in b is 3

Enter a 4 digit number

3456

3456

3456

3456

003456

Enter two floating point number

34.3

12.4

Enter the field width

4

Enter the precision

3

34.300

3.430000e+01

0.00

5.41e-315

Enter a string

cprogramming

cprogramming

cprogrammi

cprog

cprogramming

cprogramming

**PROGRAM NO: 2**

**AIM: Calculate simple interest using typedef.**

**ALGORITHM**

Step 1 : start

Step 2 : give a new name to int data type using typedef statement

Step 3 : read value of principal amount p, rate of intrest r, and number of years n

Step 4 : Find the simple interest by multiplying p\*n\*r and divide them by 100.

Step 5 : Print the value of simple interest.

Step 6 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

typedeffloat intrest;

intrest p,n,r,in;

printf("Enter the value of p,n,r:\n");

scanf("%f %f %f",&p,&n,&r);

in=(p\*n\*r)/100;

printf("%f\n",in);

}

**OUTPUT:**

Enter the value of p,n,r:

12

3

45

16.200001

**PROGRAM NO: 3**

**AIM: Find grade of a student using else if ladder.**

**ALGORITHM**

Step 1 : start

Step 2 : read values of m1,m2and m3 from user

Step 3 : Find average by add m1,m2 and m3 then divide by 3

Step 4: if average is greater than 90 then print A+ else go to step 5

Step 5 : if average greater than or equal to 80 and less than 90 print B+ else go to Step 6

Step 6 : if average is greater than or equal to 70 and less than 80 print C+ else go to step 7

Step 7 : if average is greater than or equal to 60 and less than 70 print D+ else go to step 8

Step 8 : Print ‘Fail ‘ and go to step 9

Step 9 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int m1,m2,m3,avg;

printf("Enter three marks\n");

scanf("%d%d%d",&m1,&m2,&m3);

avg=(m1+m2+m3)/3;

if(avg>=90)

printf("A+\n");

elseif(avg>=80&&avg<90)

printf("B+\n");

elseif(avg>=70&&avg<80)

printf("C+\n");

elseif(avg>=60&&avg<70)

printf("D+\n");

else

printf("fail\n");

}

**OUTPUT:**

Enter three marks

34

45

50

fail

Enter three marks

67

89

70

C+

**PROGRAM NO: 4**

**AIM: Simple calculator using switch case statement.**

**ALGORITHM**

Step 1 : start

Step 2 : read values of a and b from user

Step 3 : print menu and select choice ch

Step 4 : if ch equal to 1 go to step 5 else go to step 6

Step 5 : calculate s equal to a+b go to step 17

Step 6 : if ch equal to 2 go to step 7 else go to step 8

Step 7 : calculate s equal to a-b go to step 17

Step 8 : if ch equal to 3 go to step 9 else go to step 10

Step 9 : calculate s equal to a\*b go to step 17

Step 10 : if ch equal to 4 go to step 11 else go to step 13

Step 11 : if b equal to 0 print “Error” else go to 12

Step 12 : calculate s equal to a/b go to step 17

Step 13 : if ch equal to 5 go to step 14 else go to step 15

Step 14 : calculate s equal to a%b go to step 15

Step 15 : if ch equal to 6 go to step 16 else go to step 17

Step 16 : print “Exited”

Step 17 : if ch less than or equal to 5 go to step 3 else go to step 18

Step 18 : print “Invalid choice” go to step 3

Step 19 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int a,b,c,s,m;

do

{

printf("1.ADDITION\n");

printf("2.DIFFERENCE\n");

printf("3.MULTIPLICATION\n");

printf("4.DIVISION\n");

printf("5.MODULAS\n");

printf("Enter your choice:\n");

scanf("%d",&s);

printf("Enter two numbers:\n");

scanf("%d%d",&a,&b);

switch(s)

{

case 1:

c=a+b;

printf("SUum = %d",c);

break;

case 2:

c=a-b;

printf("Difference = %d",c);

break;

case 3:

c=a\*b;

printf("Product = %d",c);

break;

case 4:

c=a/b;

printf(" Division = %d",c);

break;

case 5:

c=a%b;

printf(" Modulas = %d",c);

break;

default:printf("invalid:");

}

printf("\nDo you want to continueee.....(1/0)\n");

scanf("%d",&m);

}while(m==1);

}

**OUTPUT:**

1.ADDITION

2.DIFFERENCE

3.MULTIPLICATION

4.DIVISION

5.MODULAS

Enter your choice:

1

Enter two numbers:

6

8

SUum = 14

Do you want to continueee.....(1/0)

1

1.ADDITION

2.DIFFERENCE

3.MULTIPLICATION

4.DIVISION

5.MODULAS

Enter your choice:

2

Enter two numbers:

8

5

Difference = 3

Do you want to continueee.....(1/0)

1

1.ADDITION

2.DIFFERENCE

3.MULTIPLICATION

4.DIVISION

5.MODULAS

Enter your choice:

3

Enter two numbers:

6

8

Product = 48

Do you want to continueee.....(1/0)

1

1.ADDITION

2.DIFFERENCE

3.MULTIPLICATION

4.DIVISION

5.MODULAS

Enter your choice:

4

Enter two numbers:

8

2

Division = 4

Do you want to continueee.....(1/0)

1

1.ADDITION

2.DIFFERENCE

3.MULTIPLICATION

4.DIVISION

5.MODULAS

Enter your choice:

5

Enter two numbers:

10

40

Modulas = 10

Do you want to continueee.....(1/0)

0

**PROGRAM NO: 5**

**AIM: Find Armstrong numbers within a range using while loop.**

**ALGORITHM**

Step 1 : start

Step 2 : read value of l from user

Step 3 : set i=l

Step 4 : Repeat while number less than or equal to l

Step 5 : set n=I and s=0

Step 6 : Repeat while n>0

Step 7 : calculate a=n%10

Step 8 : calculate p=d\*d\*d ,s=s+p and n=n/10.

Step 9 : check s==1 if it is true go to step 10

Step 10 : print value of i

Step 11 : increment I by 1 and go to step 4

Step 12 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int l,n,d,i;

printf("Enter a limit:\n");

scanf("%d",&l);

printf("Armstrong numbers are:\n");

i=1;

while(i<=l)

{

int p,s;

n=i;

s=0;

while(n>0)

{

d=n%10;

p=d\*d\*d;

s=s+p;

n=n/10;

}

if(s==i)

{

printf("%d\n",i);

}

i++;

}

}

**OUTPUT:**

Enter a limit:

1000

Armstrong numbers are:

1

153

370

371

407

**PROGRAM NO: 6**

**AIM: Calculate sum of a digit using do while loop.**

**ALGORITHM**

Step1 :Start

Step 2: Read value of n from user

Step 3: find d=n%10

Sum=sum+d

n=n/10

Step 4 : Repeat step 3 until n>0

Step 5 : print value of sum

Step 6 : Stop

**PROGRAM**

#include<stdio.h>

void main()

{

int n,sum=0,d;

printf("Enter a number\n");

scanf("%d",&n);

do

{

d=n%10;

sum=sum+d;

n=n/10;

}

while(n>0);

printf("sum=%d\n",sum);

}

**OUTPUT**

Enter a number

235

sum=10

**PROGRAM NO: 7**

**AIM: Find perfect numbers within a range using for loop.**

**ALGORITHM**

Step 1 : start

Step 2 : read value of r from user

Step 3 : set i equal to 1

Step 4 : if i less than or equal to r then go to step 5else go to step 14

Step 5 : Set s equal to 0

Step 6 : Set j equal to 1

Step 7 : if j less than or equal to i/2 then go to step 8 else go to step 13

Step 8 :check i%j equal to 0 then go to step 9 else go to

Step 9 : calculate s equal to s+j

Step 10 : check I equal to s then go to step 11

Step 11 : print the value of i

Step 12 : j equal to j + 1 go to step 7

Step 13 : i equal to i + 1 go to step 4

Step 14 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int i,j,s,r;

printf("Enter the limit\n");

scanf("%d",&r);

printf("Perfect are numbers are\n");

for(i=1;i<=r;i++)

{

s=0;

for(j=1;j<=i/2;j++)

{

if(i%j==0)

{

s=s+j;

}

}

if(i==s)

{

printf("%d \n",i);

}

}

}

**OUTPUT:**

Enter the limit

1000

Perfect are numbers are

6

28

496

**PROGRAM NO: 8**

**AIM: Implementing the break and continue statement.**

**ALGORITHM**

Step 1 : start

Step 2 : read value of n from user

Step 3 : set i equal to 0

Step 4 : if i greater than n then go to step 5 else go to step 11

Step 5 : read value of a from user

Step 6 : if a equal to 9999 go to step 11 else go to step 7

Step 7 : if a greater than 0 go to step 10 else go to step 8

Step 8 : calculate s equal to sqrt(a)

Step 9 : print value of a and s

Step 10 : i equal to i+1 go to step 4

Step 11 : stop

**PROGRAM**

#include<stdio.h>

#include<math.h>

void main()

{

int n,a,i;

float s;

printf("Enter the limit\n");

scanf("%d",&n);

printf("Enter the numbers\n");

for(i=0;i<n;i++)

{

scanf("%d",&a);

if(a==9999)

{

break;

}

if(a<0)

{

continue;

}

s=sqrt(a);

printf("Square root of %d is %f\n",a,s);

}

}

**OUTPUT:**

Enter the limit

2

Enter the numbers

9999

Enter the limit

2

Enter the numbers

25

Square root of 25 is 5.000000

3

Square root of 3 is 1.732051

**PROGRAM NO: 9**

**AIM: Average of n numbers using goto statement**.

**ALGORITHM**

Step 1 : start

Step 2 : read the value of n from user

Step 3 : set i equal to 0

Step 4 : if i greater than n then go to step 5 else go to step 9

Step 5 : read the value of p from user

Step 6 : if p less than 0 go to step 9 else go to step 7

Step 7 : calculate s equal to s+p

Step 8 : i equal to i + 1 go to step 4

Step 9 : calculate avg equal to (s/(i))

Step 10 : print the value of avg

Step 11 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int n,p,i,s=0;

float avg;

printf("Enter limit\n");

scanf("%d",&n);

printf("Enter the elements\n");

for(i=0;i<n;i++)

{

scanf("%d",&p);

if(p<0)

{

goto label;

}

s=s+p;

}

label:

avg=(s/(i));

printf("Average=%f",avg);

}

**OUTPUT:**

Enter limit

5

Enter the elements

5

4

8

10

5

Average=6.00000

**PROGRAM NO: 10**

**AIM: Simple function using pass by value.**

**ALGORITHM**

Step 1 : start

Step 2 : Set p equal to 0

Step 3 : Read value of a from user

Step 4 : Set p equal to a call function prime(a)

Step 5: if p equal to 1 then print ‘number is prime’ else goto step6

Step 6 : Print ‘Number is not prime’

Step 7 : Stop

Function prime(a)

Step 1 : start

Step 2 : set p=0

Step 3 : set i=2

Step 4 : If i lessthan or equal to a-1 yhen goto step 5 else goto step7

Step 5 : check a%i equal to 0 then goto step6

Step 6 : return 0 to function

Step 7 : check I equal to a return 1 to function

Step 8:Stop

**PROGRAM**

#include<stdio.h>

int prime(int);

void main()

{

int a,i,p=0;

printf("Enter a number\n");

scanf("%d",&a);

p=prime(a);

if(p==1)

printf("%d is a prime number\n ",a);

else

printf("%d is not a prime number\n",a);

}

int prime(int a)

{

int i,p=0;

for(i=2;i<=a-1;i++)

{

if(a%i==0)

return 0;

}

if(i==a)

return 1;

}

**OUTPUT:**

Enter a number

8

8 is not a prime number

Enter a number

0

0 is not a prime number

Enter a number

1

1 is not a prime number

Enter a number

7

7 is a prime number

**PROGRAM NO: 11**

**AIM: Simple function using pass by reference.**

**ALGORITHM**

Step 1 : start

Step 2 : read values of a and b from user

Step 3 : print values of a and b

Step 4 : call function swap(&a,&b)

Step 5 : Print values of a and b

Step 6 : Stop

**Function swap(\*a,\*b)**

Step 1 : start

Step 2 : set temp equal to \*a

Step 3 : set \*a equal to \*b

Step 4 : set \*b equal to temp

Step 5 : Stop

**PROGRAM**

#include<stdio.h>

void swap(int \*,int \*);

int a,b;

void main()

{

printf("Enter two values\n");

scanf("%d%d",&a,&b);

printf("values before swapping\n");

printf("a=%d\tb=%d\n",a,b);

swap(&a,&b);

printf("values after swapping\na=%d\tb=%d\n",a,b);

}

void swap(int \*a,int \*b)

{

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

}

**OUTPUT:**

Enter two values

45

99

values before swapping

a=45 b=99

values after swapping

a=99 b=45

**PROGRAM NO: 12**

**AIM: To find Fibonacci series by using recursive function.**

**ALGORITHM**

Step 1 : start

Step 2 : set a=0,b=1 and t=3

Step 3 : read the value of n from user

Step 4 : Print values of a and b

Step 5: set f equal to call function fib(a,b)

Step 6 : stop

**Function fact(n)**

Step 1 : start

Step 2 : if t less than or equal to n then go to step 3

Step 3 : Find C=a+b

Step 4 : Print value of c

Step 5 : Increment value of 3 by 1

Step 6 : Call function fib(a,b)

Step 7 : return f

Step 8 : Stop

**PROGRAM**

#include<stdio.h>

int fib(int,int);

int t=3,n;

int main()

{

int a=0,b=1,f,c;

printf("Enter the Limit\n");

scanf("%d",&n);

printf("Fibonacci series\n");

printf("%d\n%d\n",a,b);

f=fib(a,b);

}

int fib(int a,int b)

{

int c,f;

if(t<=n)

{

c=a+b;

printf("%d\n",c);

t++;

f=fib(b,c);

return f;

}

}

**OUTPUT:**

Enter the Limit

10

Fibonacci series

0

1

1

2

3

5

8

13

21

34

**PROGRAM NO: 13**

**AIM: Find sum and average of numbers in an array.**

**ALGORITHM**

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0,s equal to 0

Step 4: if i less than n then go to step 5 else go to step 7

Step 5: read the value of a[i] from user

Step 6: i equal to i+1 go to step 4

Step 7: set s equal to call function fun()

Step 8: print the value of s

Step 9: stop

**Function fun()**

Step 1: start

Step 2: set i equal to 0

Step 3: if i less than n then go to step 6 else go to step 9

Step 4: calculate sum equal to sum+a[i]

Step 5: i equal to i+1 go to step 3

Step 6: Print value of sum

Step 7: calculate sum divide by n and value assign to avg

Step 8: return value avg

Step 7: stop

**PROGRAM**

#include<stdio.h>

float fun();

int a[25],i,n;

void main()

{

float s;

printf("Enter a limit");

scanf("%d",&n);

printf("Enter numbers\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

s=fun();

printf("Avg=%f",s);

}

float fun()

{

int sum=0;

float avg=0;

for(i=0;i<n;i++)

{

sum=sum+a[i];

}

printf("Sum=%d\n",sum);

avg=sum/n;

return avg;

}

**OUTPUT:**

Enter limit 5

Enter numbers

45

68

46

87.6

Sum=246.000000

Avg=49.200001

**PROGRAM NO: 14**

**AIM: Largest and smallest element in an array.**

**ALGORITHM**

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

Step 4: if i less than n then go to step 5 else go to step 7

Step 5: read the value of a[i] from user

Step 6: i equal to i+1 got to step 4

Step 7: call function fun()

Step 8: stop

**Function fun()**

Step 1: start

Step 2: set large equal to 0

Step 3: set i equal to 0

Step 4: if i less than n then go to step 5 else go to step 10

Step 5: if a[i] greater than large step 6 else got to step 7

Step 6: set large equal to a[i]

Step 7: if a[i] less than small go to step 8 else got to step 9

Step 8: set s equal to a[i]

Step 9: i equal to i+1 go to step 4

Step 10: print the value of large

Step 11: print the value of small

Step 12: stop

**PROGRAM**

#include<stdio.h>

void fun();

int a[10],n,i;

void main()

{

printf("Enter limit:\n");

scanf("%d",&n);

printf("Enter numbers:\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

fun();

}

void fun()

{

int large=0,small;

for(i=0;i<n;i++)

{

if(a[i]>large)

{

large=a[i];

}

if(a[i]<small)

{

small=a[i];

}

}

printf("Largest number is=%d\n",large);

printf("Smallest number is=%d\n",small);

}

**OUTPUT:**

Enter limit:

4

Enter numbers:

3

5

33

4

Largest number is=33

Smallest number is=3

**PROGRAM NO: 15**

**AIM: Linear search using function.**

**ALGORITHM**

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

Step 4: if i less than n then go to step 5 else go to step 7

Step 5: read the value of a[i] from user

Step 6: i equal to i+1 got to step 4

Step 7: call function search ()

Step 8: if f equal to 0 print element not found

Step 9: Stop

**Function search()**

Step 1: start

Step 2: read the value of s from user

Step 3: set i equal to 0

Step 4: if i less than n then go to step 5 else go to step 8

Step 5: if a[i] equal to s then go to step 6 else go to step 7

Step 6: print the value of i+1 go to step 10

Step 7: Set flag value equal to 1

Step 8: i equal to i+1 got to step 4

Step 9: Return value of f

Step 10: stop

**PROGRAM**

#include<stdio.h>

int search();

int f=0,n,a[25],p,i,l;

void main()

{

printf("Enter the limit\n");

scanf("%d",&n);

printf("enter the numbers\n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

search();

if(f==0)

printf("not found\n");

}

int search()

{ int s;

printf("enter the number to be searched\n");

scanf("%d",&s);

for(i=0;i<n;i++)

{

if(a[i]==s)

{

printf("number found at %d position\n",i+1);

f=1;

}

}

return f;

}

**OUTPUT:**

Enter the limit

5

enter the numbers

6

5

7

4

8

enter the number to be searched

3

not found

Enter the limit

5

enter the numbers

4

6

3

3

3

enter the number to be searched

3

number found at 3 position

number found at 4 position

number found at 5 position

**PROGRAM NO: 16**

**AIM: Selection sort using function.**

**ALGORITHM**

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

Step 4: if i less than n then go to step 5 else go to step 7

Step 5: read the value of a[i] from user

Step 6: i equal to i+1 got to step 4

Step 7: call function selection (a,n)

Step 8: stop

**Function selection(a[ ],n)**

Step 1: start

Step 2: set i equal to 0

Step 3: if i less than n then go to step 3 else go to step 12

Step 4: set j equal to i+1

Step 5: if j less than n-1 then go to step 6 else go to step 11

Step 6: if ( a[i] > a[j] ) go to step 7 else got to step 10

Step 7: set temp equal to a[i]

Step 8: set a[i] equal to a[j]

Step 9: set a[j] equal to temp

Step 10: j equal to j+1 go to step 5

Step 11: i equal to i+1 go to step 3

Step 12: set i equal to 0

Step 13: if i less than n then go to step 14 else go to step 16

Step 14: print the value of a[i]

Step 15: i equal to i+1 got to step 13

Step 16: stop

**PROGRAM**

#include<stdio.h>

void selection(int [],int);

int i,j;

void main()

{

int a[10],n;

printf("Enter limit\n");

scanf("%d",&n);

printf("Enter numbers\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

selection(a,n);

}

void selection(int a[],int n)

{

int temp;

for(i=0;i<n;i++)

{

for(j=i+1;j<n;j++)

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

printf("sorted numbers are\n");

for(i=0;i<n;i++)

{

printf(" %d\n",a[i]);

}

}

**OUTPUT:**

Enter limit

8

Enter numbers

34

1

78

90

4

3

6

99

sorted numbers are

1

3

4

6

34

78

90

99

**PROGRAM NO: 17**

**AIM: Bubble sort using function.**

**ALGORITHM**

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

Step 4: if i less than n then go to step 5 else go to step 7

Step 5: read the value of a[i] from user

Step 6: i equal to i+1 got to step 4

Step 7: call function bubble(a,n)

Step 8: stop

**Function bubble(a[ ],n)**

Step 1 : start

Step 2 : set i equal to 0

Step 3 : if i less than n then go to step 3 else go to step 12

Step 4 : set j equal to 0

Step 5 : if j less than n-i-1then go to step 6 else go to step 11

Step 6 : if ( a[j] > a[j+1] ) go to step 7 else got to step 10

Step 7 : set temp equal to a[j]

Step 8 : set a[j] equal to a[j+1]

Step 9 : set a[j+1] equal to temp

Step 10: j equal to j+1 go to step 5

Step 11: i equal to i+1 go to step 3

Step 12: set i equal to 0

Step 13: if i less than n then go to step 14 else go to step 16

Step 14: print the value of a[i]

Step 15: i equal to i+1 got to step 13

Step 16: stop

**PROGRAM**

#include<stdio.h>

void bubble(int [],int);

int i,j;

void main()

{

int a[10],n;

printf("Enter limit\n");

scanf("%d",&n);

printf("Enter numbers\n");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

bubble(a,n);

}

void bubble(int a[],int n )

{

int temp;

for(i=0;i<n;i++)

{

for(j=0;j<(n-i-1);j++)

{

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("sorted numbers are\n");

for(i=0;i<n;i++)

{

printf(" %d\n",a[i]);

}

}

**OUTPUT:**

Enter limit

7

Enter numbers

8

3

2

78

45

21

4

sorted numbers are

2

3

4

8

21

45

78

**PROGRAM NO: 18**

**AIM: Matrix multiplication using function**.

**ALGORITHM**

Step 1 : start

Step 2 : read the value of r1 and c1 from user

Step 3 : read the value of r2 and c2 from user

Step 4 : if r1 not equal to c2 then print multiplication not possible else go tostep 5

Step 5 : set i equal to 0

Step 6 : if i less than r2 then go to step 7 else go to step 12

Step 7 : set j equal to 0

Step 8 : if j less than c2 then go to step 9 else go to step 11

Step 9 : read the value of a[i][j] from user

Step 10: j equal to j+1 got to step 8

Step 11: i equal to i+1 go to step 6

Step 12: set i equal to 0

Step 13: if i less than r1 then go to step 14 else go to step 19

Step 14: set j equal to 0

Step 15: if j less than c2 then go to step 16 else go to step 18

Step 16: read the value of b[i][j] from user

Step 17: j equal to j+1 got to step 15

Step 18: i equal to i+1 go to step 13

Step 19: Print two metrices a[i][j] and b[i][j]

Step 20: Call function mul(a,b,r1,c2)

Step 21: stop

**Function mul()**

Step 1 : start

Step 2 : set i equal to 0

Step 3 : if i less than r1 then go to step 4 else go to step 13

Step 4 : set j equal to 0

Step 5 : if j less than c2 then go to step 6 else go to step 8

Step 6 : set m[i][j] equal to 0

Step 7 : set k equal to 0

Step 8 : if k less than r1 then go to step 9 else go to step 11

Step 9 : calculate m[i][j] equal to m[i][j]+(a[i][k]\*b[k][j])

Step 10: k equal to k+1 go to step 8

Step 11: j equal to j+1 go to step 5

Step 12: i equal to i+1 go to step 3

Step 13: set i equal to 0

Step 14: if i less than m then go to step 15 else go to step 20

Step 15: set j equal to 0

Step 16: if j less than n then go to step 17 else go to step 19

Step 17: print the value of m[i][j]

Step 18: j equal to j+1 go to step 16

Step 19: i equal to i+1 go to step 14

Step 20: stop

**PROGRAM**

#include<stdio.h>

void mul(int [][10],int [][10],int ,int);

inti,j,r1,c1,r2,c2,k;

void main()

{

int a[10][10],b[10][10];

printf("Enter row and column of first matrix\n");

scanf("%d%d",&r1,&c1);

printf("Enter row and column of second matrix\n");

scanf("%d%d",&r2,&c2);

if(r1!=c2)

printf("not possible\n");

else

{

printf("enter the elements of first matrix\n");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

scanf("%d",&a[i][j]);

}

}

printf("enter the elements of second matrix\n");

for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++)

{

scanf("%d",&b[i][j]);

}

}

printf("FIRST MATRIX\n");

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

printf("%d\t",a[i][j]);

}

printf("\n");

}

printf("SECOND MATRIX\n");

for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++)

{

printf("%d\t",b[i][j]);

}

printf("\n");

}

mul(a,b,r1,c2);

}

}

void mul(int a[][10],int b[][10],int r1,int c2)

{

int m[10][10];

printf("multiplication table\n");

for(i=0;i<r1;i++)

{

for(j=0;j<c2;j++)

{

m[i][j]=0;

for(k=0;k<r1;k++)

{

m[i][j]=m[i][j]+(a[i][k]\*b[k][j]);

}

printf("%d\t",m[i][j]);

}

printf("\n");

}

}

**OUTPUT**

Enter row and column of first matrix

2 2

Enter row and column of second matrix

2 2

enter the elements of first matrix

3 4 5 7

enter the elements of second matrix

4 5 6 8

FIRST MATRIX

3 4

5 7

SECOND MATRIX

4 5

6 8

multiplication table

36 47

62 81

Enter row and column of first matrix

2 3

Enter row and column of second matrix

3 2

enter the elements of first matrix

3 4 6 6 7 8

enter the elements of second matrix

2 4 5 6 7 8

FIRST MATRIX

3 4 6

6 7 8

SECOND MATRIX

2 4

5 6

7 8

multiplication table

26 36

47 66

Enter row and column of first matrix

2 3

Enter row and column of second matrix

2 3

not possible

**PROGRAM NO: 19**

**AIM: Sum of diagonal elements of a matrix.**

**ALGORITHM**

Step 1 : start

Step 2 : read the value of r and c from user

Step 3 :if r not equal to c then print not possible else goto step 4

Step 4 : set i equal to 0

Step 5 : if i less than r then go to step 6 else go to step 11

Step 6 : set j equal to 0

Step 7 : if j less than c then go to step 8 else go to step 10

Step 8 : read the value of a[i][j] from user

Step 9 : j equal to j+1 got to step 7

Step 10: i equal to i+1 go to step 5

Step 11: Print two metrix a[i][j]

Step 12: Call function dioganall(a,r,c)

Step 13: stop

**Function diagonal()**

Step 1 : start

Step 2 : set sum1 and sum2 equal to 0

Step 3 : set i equal to 0

Step 4 : if i less than r then go to step 5 else go to step 13

Step 5 : set j equal to 0

Step 6 : if j less than c2 then go to step 7 else go to step 8

Step 7 : if I equal to j then go to step 8

Step 8 : Find sum1+a[i][j] and store to sum1

Step 9 : if (i+j) equal to (r-1) then goto step 10

Step 10: find sum2+a[i][j] and store to sum2

Step 11: j equal to j+1 got to step 6

Step 12: i equal to i+1 go to step 4

Step 13: print the value of sum1 as sum of trace or principal diagonal elements

Step 14: print the value of sum2 as sum of diagonal elements

Step 15:stop

**PROGRAM**

#include<stdio.h>

void dioganal(int [][10],int,int);

inti,j;

void main()

{

int a[10],r,c;

printf("Enter row and column of matrix\n");

scanf("%d%d",&r,&c);

if(r!=c)

printf("Not possible\n");

else

printf("enter the elements of matrix\n");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

scanf("%d",&a[i][j]);

}

}

printf("matrix is\n");

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

printf("%d\t",a[i][j]);

}

printf("\n");

}

dioganal(a,r,c);

}

void dioganal(int a[][10],int r,int c)

{

int sum1=0,sum2=0;

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

if(i==j)

{

sum1=sum1+a[i][j];

}

if((i+j)==(r-1))

{

sum2=sum2+a[i][j];

} }

}

printf("Sum of Trace or principl dioganal elements is=%d\n",sum1);

printf("Sum of off dioganal elements= %d\n",sum2);

}

**OUTPUT**

Enter row and column of matrix

2

2

enter the elements of matrix

3

4

5

6

matrix is

3 4

5 6

Sum of Trace or principl dioganal elements is=9

Sum of off dioganal elements= 9

Enter row and column of matrix

2

3

Not possible

**PROGRAM NO: 20**

**AIM: String manipulations.**

**ALGORITHM**

Step 1 : start

Step 2 : read the value of s from user

Step 3 : find the length of string s then go to step 4

Step 4 : print the string length

Step 5 : read value of s1 and s2 from user

Step 6 : copy the string s2 to s1

Step 7 :print the value of s1

Step 8 : read value of s3 and s4from user

Step 9 : compare s3 and s4 if it is equal to 0 then go to step 10 else go to step 11

Step 10 : print ‘strings are equal’

Step 11 : print ‘strings are not equal

Step 12 : concatenate strings s3 and s4

Step 13 : print value of s3

Step 14: stop

**PROGRAM**

#include<stdio.h>

#include<string.h>

void main()

{

char s[20],s1[10],s2[10],s3[10],s4[10];

int n,h;

printf("Enter the string:\n");

scanf("%s",s);

n=strlen(s);

printf("\*\*\*\*LENGTH\*\*\*\*\n");

printf("Length of string using strlen: %d\n",n);

printf("Enter two strings:\n");

scanf("%s%s",s1,s2);

strcpy(s1,s2);

printf("\*\*\*\*\*\*COPY\*\*\*\*\*\n");

printf("Copied string is:%s\n",s1);

printf("Enter two strings:\n");

scanf("%s%s",s3,s4);

printf("\*\*\*\*\*COMPARE\*\*\*\*\n");

h=strcmp(s3,s4);

if(h==0)

{

printf("Two strings are equal\n");

}

else

{

printf("Two strings are not equal\n");

}

strcat(s3,s4);

printf("\*\*\*\*\*CONCATENATION\*\*\*\*\n");

printf("The concatenated string:%s\n",s3);

}

**OUTPUT:**

Enter the string:

Newdelhi

\*\*\*\*LENGTH\*\*\*\*

Length of string using strlen: 8

Enter two strings:

Newdelhi Bangloor

\*\*\*\*\*\*COPY\*\*\*\*\*

Copied string is:Bangloor

Enter two strings:

kannur mattannur

\*\*\*\*\*COMPARE\*\*\*\*

Two strings are not equal

\*\*\*\*\*CONCATENATION\*\*\*\*

The concatenated string:kannurmattannur

**PROGRAM NO: 21**

**AIM: Find length of a string using pointer.**

**ALGORITHM**

Step 1: start

Step 2: read the value of str from user

Step 3: set p equal to str

Step 4: if \*p not equal to ’\0’ then go to step 5 else go to s

Step 5: c equal to c+1 go to step 6

Step 6: p equal to p+1 go to step 4

Step 7: print value of c

Step 8: stop

**PROGRAM**

#include<stdio.h>

void main()

{

char str[20],\*p;

int c=0;

printf("Enter a string\n");

scanf("%s",str);

for(p=str;\*p!='\0';p++)

{

c++;

}

printf("Length=%d\n",c);

}

**OUTPUT:**

Enter a string

programming

Length=11

**PROGRAM NO: 22**

**AIM: Sort list of names using array of pointer.**

**ALGORITHM**

Step 1 : start

Step 2 : read the value of n from user

Step 3 : set i equal to 0

Step 4 : if i less than n then go to step 5 else go to step 10

Step 5 : read the value of s[i] from user

Step 6 : i equal to i+1 got to step 4

Step 7 : set i equal to 0

Step 8 : if i less than n then go to step 9 else go to step 16

Step 9 : set j equal to i+1

Step 10: if j less than n then go to step 11 else go to step 15

Step 11: compare &s[i ]and &s[j] if it is greater than 0 then go to step 12 else go to step15

Step 12: set temp equal to s[i]

Step 13: set s[i] equal to s[j]

Step 14: set s[j] equal to temp

Step 15: j equal to j+1 go to step 10

Step 16: i equal to i+1 go to step 8

Step 17: print the value of s[i]

Step 18: stop

**PROGRAM**

#include<stdio.h>

void main()

{

char \*s[20],\*temp;

int i,j,n;

printf("Enter limits:\n");

scanf("%d",&n);

printf("Enter names:\n");

for(i=0;i<n;i++)

{

scanf("%s",&s[i]);

//gets(s[i]);

}

for(i=0;i<n;i++)

{

for(j=i+1;j<n;j++)

{

if(strcmp(&s[i],&s[j])>0)

{

temp=s[i];

s[i]=s[j];

s[j]=temp;

}

}

}

printf("Sorted names are:\n");

for(i=0;i<n;i++)

{

printf("%s\n",&s[i]);

}

}

**OUTPUT**

Enter limits:

7

Enter names:

Sana

Kiran

Ajay

Arun

Jithesh

Kumar

Aleena

Sorted names are:

Ajay

Aleena

Arun

Jithesh

Kiran

Kumar

Sana

**PROGRAM NO: 23**

**AIM: Implementation of pointer to pointer**

**ALGORITHM**

Step 1 : start

Step 2 : set p1 equal to &a

Step 3 : set p2 equal to &p1

Step 4 : set q1 equal to &b

Step 5 : set q2 equal to &q1

Step 7 : read the value of a and b

Step 8 : calculate c equal to \*\*p2 + \*\*q2

Step 9 : print the value of c

Step 10 : calculate c equal to \*\*p2 - \*\*q2

Step 11 : print the value of c

Step 12 : calculate c equal to \*\*p2 \* \*\*q2

Step 13 : print the value of c

Step 14 : calculate c equal to \*\*p2 / \*\*q2

Step 15 : print the value of c

Step 16 : stop

**PROGRAM**

#include<stdio.h>

void main()

{

int a,b,c,\*p1,\*\*p2,\*q1,\*\*q2;

p1=&a;

p2=&p1;

q1=&b;

q2=&q1;

printf("Enter two numbers\n");

scanf("%d%d",&a,&b);

c=\*\*p2+\*\*q2;

printf("sum=%d\n",c);

c=\*\*p2-\*\*q2;

printf("Difference=%d\n",c);

c=\*\*p2\*\*\*q2;

printf("product=%d\n",c);

c=\*\*p2/ \*\*q2;

printf("division=%d\n",c);

}

**OUTPUT:**

Enter two numbers

12

6

sum=18

Difference=6

product=72

division=2

**PROGRAM NO: 24**

**AIM: Implementation of enumerated types.**

**ALGORITHM**

Step 1 : start

Step 2 : create enumerated data type subjects

Step 3 : create enumerated variable sub

Step 4 :set sub equal to Malayalam

Step 5 : if sum less than or equal to mathematics then go to step 6 else goto7

Step 6 : read value of marks from user

Step 7 : sub equal to sub+1

Step 8: print values of marks of each subject

Step 9: stop

**PROGRAM**

#include<stdio.h>

enum subjects

{

malayalam,

english,

computer,

maths

}sub;

void main()

{

int marks[40];

//enum subjects sub;

printf("Enter marks of 4 subjects\n");

for(sub=malayalam;sub<=maths;sub++)

{

scanf("%d",&marks[sub]);

}

printf("marks in different subjects are\n");

printf("malayalam=%d\nEnglish=%d\ncomputer=%d\nmaths=%d\n",marks[malayalam],marks[english],marks[computer],marks[maths]);

}

**OUTPUT:**

Enter marks of 4 subjects

45

67

36

46

marks in different subjects are

malayalam=45

English=67

computer=36

maths=46

**PROGRAM NO: 25**

**AIM: Implementation of bifields.**

**ALGORITHM**

Step 1 : start

Step 2 : create a structure sample

Step 3 : create structure variable e

Step 4 : set e.a equal to 4

Step 5 : set e.b equal to 1

Step 6 : print the value of e.a

Step 7 : print the value of e.b

Step 8 : calculate c equal to e.a + e.b

Step 9 : print the value ofc

Step 10 : calculate c equal to e.a - e.b

Step 11: print the value of c

Step 12 : calculate c equal to e.a \* e.b

Step 13 : print the value of c

Step 14 : stop

**PROGRAM**

#include<stdio.h>

struct sample

{

unsignedint a:3;

unsignedint b:2;

}e;

void main()

{

int c;

e.a=4;

e.b=1;

printf("%d\t",e.a);

printf("%d\t\n",e.b);

c=e.a+e.b;

printf("sum=%d\n",c);

c=e.a-e.b;

printf("difference=%d\n",c);

c=e.a\*e.b;

printf("product=%d\n",c);

}

**OUTPUT:**

4 1

sum=5

difference=3

product=4

**PROGRAM NO: 26**

**AIM: Employee details using structure.**

**ALGORITHM**

Step 1 : start

Step 2 : create a structure emp

Step 3 : create structure variable e[10]

Step 4 : print menu and select choice s

Step 5 : read the value of ch from user

Step 6 : if s equal to 1 call function read() and go to step 12

Step 7 : if s equal to 2 call function display() and go to step 12

Step 8 : if s equal to 3 call function search() and go to step 12

Step 9 : if s equal to 4 call function del() and go to step 12

Step 10 : if s equal to 5 go to step 12 else go to step 11

Step 11 : print “Invalid”

Step 12 : if s less than or equal to 4 go to step 3 else go to step 13

Step 13 : stop

**Function read()**

Step 1 : start

Step 2 : read the value of n from user

Step 3 : set i equal to 0

Step 4 : if i less than n then go to step 5 else go to step 11

Step 5 : print the value of i+1

Step 6 : read the value of e[i].empnofrom user

Step 7 : read the value of e[i].name from user

Step 8 : read the value of e[i].salaryfrom user

Step 9 : i equal to i+1 go to step 4

Step 10 : stop

**Function display()**

Step 1 : start

Step 2 : set i equal to 0

Step 3 : if i less than n then go to step 4 else go to step 6

Step 4 : print values of e[i].empno,e[i].name,e[i].salary

Step 5 : i equal to i+1 go to step 3

Step 6 : stop

**Function search()**

Step 1 : start

Step 2 : set f equal to 0

Step 3 : read the value of m from user

Step 4 : set i equal to 0

Step 5 : if i less than n then go to step 6 else go to step 13

Step 6 : if (e[i].name,m==k) go to step 7 else go to step12

Step 7 : print the value of e[i].empno

Step 8 : print the value of e[i].name

Step 9 : print the value of e[i].salary

Step 10: i equal to i+1 go to step 5

Step 11 : if I equal to n goto step 12

Step 12 : print “ not found”

Step 13 : stop

**Function dele()**

Step 1 : start

Step 2 : read the value of k from user

Step 3 : set i equal to 0

Step 4 : if i less than n then go to step 5 else go to step 14

Step 5 : if e[i].name equal to k go to step 6 else go to step 12

Step 6 : set f equal to 1

Step 7 : set pos equal to i

Step 8 : i equal to i+1 go to step 8

Step 9:if f equal to l goto step 10 else goto step 15

Step 10:set I equalto pos

Step 11:if I lessthan n-l then goto step 12 else goto step 13

Step 12:set e[i] equal to e[i+1]

Step 13 : i equal to i+1

Step 14 : print value of empno,name and salary

Step 15 :print not found

Step 16 : stop

**PROGRAM**

#include<stdio.h>

int n,i,s,c;

char m[10],k[10];

struct emp

{

int empno;

char name[20];

float salary;

}e[10];

void read(struct emp[]);

void display(struct emp[]);

void search(struct emp[]);

void delt(struct emp[]);

void main()

{

do

{

printf("\n 1 ENTER ");

printf("\n 2 DISPLAY");

printf("\n 3 SEARCH");

printf("\n 4 DELETE");

printf("\n 5 EXIT");

printf("\nEnter your choice:\n");

scanf("%d",&s);

switch(s)

{

case 1:

read(e);

break;

case 2:

display(e);

break;

case 3:

search(e);

break;

case 4:

delt(e);

break;

default:printf("invalid\n");

}

printf("\n Do you want to continue (1/0)\t");

scanf("%d",&c);

}while(c==1);

}

void read(struct emp e[])

{

printf("Enter limits:\n");

scanf("%d",&n);

printf("Enter details:\n");

for(i=0;i<n;i++)

{

scanf("%d%s%f",&e[i].empno,e[i].name,&e[i].salary);

}

}

void display(struct emp e[])

{

printf("Details are:\n");

for(i=0;i<n;i++)

{

printf("%d\n%s\n%f\n",e[i].empno,e[i].name,e[i].salary);

}

}

void search(struct emp e[])

{

printf("Enter the name to be searched:\n");

scanf("%s",m);

for(i=0;i<n;i++)

{

if(strcmp(e[i].name,m)==0)

{

printf("Name found\n");

printf("%d\n%s\n%f\n",e[i].empno,e[i].name,e[i].salary);

break;

}

}

if(i==n)

printf("Name not found\n");

}

void delt(struct emp e[])

{

int pos,f=0;

printf("Enter the name to be deleted\n");

scanf("%s",k);

for(i=0;i<n;i++)

{

if(strcmp(e[i].name,k)==0)

{

f=1;

pos=i;

break;

}

}

if(f==1)

{

for(i=pos;i<n-1;i++)

{

e[i]=e[i+1];

}

printf("\n") ;

for(i=0;i<n-1;i++)

{

printf("%d\n%s\n%f\n",e[i].empno,e[i].name,e[i].salary);

}

}

else

printf("not found\n");

}

**OUTPUT:**

1 ENTER

2 DISPLAY

3 SEARCH

4 DELETE

5 EXIT

Enter your choice:

1

Enter limits:

3

Enter details:

101 Nidhin 30000

102 Arun 25000

103 Subina 30000

Do you want to continue (1/0) 1

1 ENTER

2 DISPLAY

3 SEARCH

4 DELETE

5 EXIT

Enter your choice:

2

Details are:

101

Nidhin

30000.000000

102

Arun

25000.000000

103

Subina

30000.000000

Do you want to continue (1/0) 1

1 ENTER

2 DISPLAY

3 SEARCH

4 DELETE

5 EXIT

Enter your choice:

3

Enter the name to be searched:

Arun

Name found

102

Arun

25000.000000

Do you want to continue (1/0) 1

1 ENTER

2 DISPLAY

3 SEARCH

4 DELETE

5 EXIT

Enter your choice:

4

Enter the name to be deleted

Arun

101

Nidhin

30000.000000

103

Subina

30000.000000

Do you want to continue (1/0) 1

1 ENTER

2 DISPLAY

3 SEARCH

4 DELETE

5 EXIT

Enter your choice:

5

invalid

Do you want to continue (1/0) 0

**PROGRAM NO: 27**

**AIM: Implementing union using function.**

**ALGORITHM**

**Function display()**

Step 1 : start

Step 2 : set i equal to 0

Step 3 : if i less than n then go to step 4 else go to step 6

Step 4 : print values of e[i].empno,e[i].name,e[i].salary

Step 5 : i equal to i+1 go to step 3

Step 6 : stop

**Function search()**

Step 1 : start

Step 2 : set f equal to 0

Step 3 : read the value of m from user

Step 4 : set i equal to 0

Step 5 : if i less than n then go to step 6 else go to step 13

Step 6 : if (e[i].name,m==k) go to step 7 else go to step12

Step 7 : print the value of e[i].empno

Step 8 : print the value of e[i].name

Step 9 : print the value of e[i].salary

Step 10: i equal to i+1 go to step 5

Step 11 : if I equal to n goto step 12

Step 12 : print “ not found”

Step 13 : stop

**Function dele()**

Step 1 : start

Step 2 : read the value of k from user

Step 3 : set i equal to 0

Step 4 : if i less than n then go to step 5 else go to step 14

Step 5 : if e[i].name equal to k go to step 6 else go to step 12

Step 6 : set f equal to 1

Step 7 : set pos equal to i

Step 8 : i equal to i+1 go to step 8

Step 9:if f equal to l goto step 10 else goto step 15

Step 10:set I equalto pos

Step 11:if I lessthan n-l then goto step 12 else goto step 13

Step 12:set e[i] equal to e[i+1]

Step 13 : i equal to i+1

Step 14 : print value of empno,name and salary

Step 15 :print not found

Step 16 : stop

**PROGRAM**

#include<stdio.h>

void student();

union stdt

{

int rollno;

char name[20];

int mark;

}s;

void main()

{

student();

}

void student()

{

printf("Enter roll number\n");

scanf("%d",&s.rollno);

printf("Student Roll:%d\n",s.rollno);

printf("Enter student name\n");

scanf("%s",s.name);

printf("student name:%s\n",s.name);

printf("Enter student mark\n");

scanf("%d",&s.mark);

printf("student mark is:%d\n",s.mark);

}

**OUTPUT:**

Enter roll number

12

Student Roll:12

Enter student name

Dileena

student name:Dileena

Enter student mark

45

student mark is:45

**PROGRAM NO: 28**

**AIM: Telephone directory using file.**

**ALGORITHM**

Step 1 : start

Step 2 : create a structure telephone

Step 3 : create a structure variable t

Step 4 : print menu and select choice ch

Step 5 : if ch equal to 1 then call function insert()

and go to step 12

Step 6 : if ch equal to 2 then call function view()

and go to step 12

Step 7 : if ch equal to 3 then call function search()

and go to step 12

Step 8 : if ch equal to 4 then call function update()

and go to step 12

Step 9 : if ch equal to 5 then call function delete()

and go to step 12

Step 10 : if ch equal to 6 go to step 13 else go to step 11

Step 11 : print “exited”

Step 12 : if ch not equal to 6 go to step 4 else go to step 13

Step 13 : stop

**Function insert()**

Step 1 : start

Step 2 : declare the file pointer fp

Step 3 : open the file with a+ mode

Step 4 : if fp equal to NULL then go to step 5 else go to step 6

Step 5 : Open the file with w mode go to step 8

Step 6 : read id,name,area and telephone number to the file

Step 7 : close the file

Step 8 : end

Step 9 : stop

**Function view()**

Step 1 : start

Step 2 : declare the file pointer fp

Step 3 : open a file with r mode

Step 4 : if fp equal to NULL then go to step 5 else go to step 6

Step 5 : Print “No records” go to step 9

Step 6 : if (fread(&t1,sizeof(t1),1,fp)==1) then go to step 7 else

go to step 8

Step 7 : Print the values of id,name,area and phone number and

go to step 6

Step 8 : close the file

Step 9 : stoP

**Function search()**

Step 1 : start

Step 2 : declare file pointer fp

Step 3 : open a file with r mode

Step 4 : if fp equal to NULL then go to step 5 else go to step 6

Step 5 : Print “No records”

Step 6 : read the value of n from user

Step 7 : if (fread(&t1,sizeof(t1),1,fp)==1) then go to step 8 else

go to step 11

Step 8 : if t.id equal to n then go to step 9 else go to step 7

Step 9 : Print the values id,name,area and phone number

Step 10 : set f equal to 1 and go to step 11

Step 11 : if f equal to 0 go to step 12 else go to step 13

Step 12 : print “ can not found”

Step 13 : close the file

Step 14 : stop

**Function update()**

Step 1 : start

Step 2 : declare the file pointer fp

Step 3 : set f equal to 0

Step 4 : open a file with r+ mode

Step 5 : set the pointer fp at first

Step 6 : set r equal to size of the file

Step 7 : if fp equal to NULL then go to step 8 else go to step 9

Step 8 : Print ”No records”

Step 9 : read the value of n from user

Step 10 : if (fread(&t1,sizeof(t1),1,fp)==1) then go to step 11

else go to step 15

Step 11 : if t.id equal to n then go to step 12 else go to step 10

Step 12 : read values of id,name,area and phone number

to the file

Step 13 : Set the file pointer to the particular point

Step 14 : set f equal to 1 go to step 10

Step 15 : if f equal to 0 then go to step 16 else go to step 17

Step 16 : print “ can not found”

Step 17 : close the file

Step 18 : stop

**Function delete()**

Step 1 : start

Step 2 : declare file pointers fp and f1

Step 3 : set f equal to 0

Step 4 : open a file with r mode in fp and w mode in f1

Step 5 : if fp equal to NULL then go to step 6 else go to step 7

Step 6 : Print “No records”

Step 7 : read the value of n from user

Step 8 : if (fread(&t1,sizeof(t1),1,fp)==1) then go to step 9 else

go to step 11

Step 9 : if t.id not equal to n go to step 10 else go to step 8

Step 10 : print the values of id,name,area and phone number

Step 11 : close the file fp

Step 12 : set k equal to current position of the pointer f1

Step 13 : close the file f1

Step 14 : open a file with r mode in f1 and w mode in fp

Step 15 : if (fread(&t1,sizeof(t1),1,fp)==1) then go to step 17 else

go to step 18

Step 16 : print the values of id,name,area and phone number

Step 17 : Write a record to the file go to step 15

Step 18 : set the pointer fp at first

Step 19 : close the files f1 , fp

Step 20 : stop

**PROGRAM**

#include<stdio.h>

void insert();

void view();

void search();

void update();

voiddelete();

struct tel

{

int id;

char name[30];

char area[20];

longint phno;

}t;

void main()

{

int ch;

do

{

printf("1.INSERT\n");

printf("2.VIEW\n");

printf("3.SEARCH\n");

printf("4.UPDATE\n");

printf("5.DELETE\n");

printf("6.EXIT\n");

printf("Enter the option:\n");

scanf("%d",&ch);

switch(ch)

{

case 1:insert();

break;

case 2:view();

break;

case 3:search();

break;

case 4:update();

break;

case 5:delete();

break;

case 6:break;

default:

printf("INVALID");

break;

}

}while(ch!=6);

}

void insert()

{

FILE \*fp;

char c;

fp=fopen("TEL","a+");

if(fp==NULL)

{

printf("No record\n enter the first record\n");

fp=fopen("TEL","w");

}

else

{ printf("Enter the Id:\n");

scanf("%d",&t.id);

c=getchar();

printf("Enter the name:\n");

scanf("%s",t.name);

printf("Enter the area:\n");

scanf("%s",t.area);

printf("Enter the phone number:");

scanf("%ld",&t.phno);

fwrite(&t,sizeof(t),1,fp);

} fclose(fp);

}

void view()

{

FILE \*fp;

unsignedlong l;

fp=fopen("TEL","r");

if(fp==NULL)

printf("File doesn't occur\n");

else

{

fseek(fp,0,2);

l=(unsignedlong)ftell(fp);

rewind(fp);

if(l<=0)

printf("No records\n");

else

{

printf("ID\tNAME\tAREA\tPh Number\n");

while(fread(&t,sizeof(t),1,fp)==1)

{

printf("%d\t%s\t%s\t%ld\t\n",t.id,t.name,t.area,t.phno);

}

}

fclose(fp);

}

}

void search()

{

FILE \*fp;

int n,f=0;

unsignedlong l;

fp=fopen("TEL","r");

if(fp==NULL)

printf("File doesn't occur\n");

else

{

fseek(fp,0,2);

l=(unsignedlong)ftell(fp);

rewind(fp);

if(l<=0)

printf("No records\n");

else

{

printf("Enter one ID you want to search:\n");

scanf("%d",&n);

while(fread(&t,sizeof(t),1,fp)==1)

{

if(t.id==n)

{

printf("ID\tNAME\tAREA\tPh Number\n");

printf("%d\t%s\t%s\t%ld\t\n",t.id,t.name,t.area,t.phno);

printf("\n");

f=1;

break;

}

}

if(f==0)

printf("Cannot found\n");

fclose(fp);

}

}

}

void update()

{

FILE \*fp;

int n,r,f=0;

char c;

unsignedlong l;

r=sizeof(t);

fp=fopen("TEL","r+");

if(fp==NULL)

printf("File doesn't occur\n");

else

{

fseek(fp,0,2);

l=(unsignedlong)ftell(fp);

rewind(fp);

if(l<=0)

printf("No records\n");

else

{

printf("Enter the ID to modified:\n");

scanf("%d",&n);

while(fread(&t,sizeof(t),1,fp)==1)

{

if(t.id==n)

{

printf("Enter the new ID:\n");

scanf("%d",&t.id);

c=getchar();

printf("Enter the new name:\n");

scanf("%s",t.name);

printf("Enter the new area:\n");

scanf("%s",t.area);

printf("Enter the new phone number:\n");

scanf("%ld",&t.phno);

fseek(fp,-r,SEEK\_CUR);

fwrite(&t,sizeof(t),1,fp);

printf("Suucefully updated\n");

f=1;

break;

}

}

if(f==0)

printf("Cannot found\n");

fclose(fp);

}

}

}

voiddelete()

{

FILE \*fp,\*f1;

int n,r,f=0,k;

unsignedlong l;

char c;

fp=fopen("TEL","r");

if(fp==NULL)

printf("File doesn't occur\n");

else

{

fseek(fp,0,2);

l=(unsignedlong)ftell(fp);

rewind(fp);

if(l<=0)

printf("No records\n");

else

{

f1=fopen("TEL1","w");

printf("Enter the ID to be deletd:\n");

scanf("%d",&n);

while(fread(&t,sizeof(t),1,fp)==1)

{

if(t.id==n)

f=1;

else

fwrite(&t,sizeof(t),1,f1);

}

fclose(fp);

k=ftell(f1);

fclose(f1);

fp=fopen("TEL","w");

f1=fopen("TEL1","r");

while(fread(&t,sizeof(t),1,f1)==1)

{

fwrite(&t,sizeof(t),1,fp);

}

if(f==1)

printf("Successfully deleted\n");

else

printf("Id not found\n");

rewind(fp);

fclose(f1);

fclose(fp);

}

}

}

**OUTPUT:**

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

1

Enter the Id:

1

Enter the name:

anu

Enter the area:

perinkary

Enter the phone number:87654321980

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

1

Enter the Id:

2

Enter the name:

amal

Enter the area:

vallithodu

Enter the phone number:34346612

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

1

Enter the Id:

3

Enter the name:

ramees

Enter the area:

panoor

Enter the phone number:78621342

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

2

ID NAME AREA Ph Number

1 anu perinkary 87654321980

2 amal vallithodu 34346612

3 ramees panoor 78621342

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

3

Enter one ID you want to search:

2

ID NAME AREA Ph Number

2 amal vallithodu 34346612

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

4

Enter the ID to modified:

2

Enter the new ID:

6

Enter the new name:

subina

Enter the new area:

kannur

Enter the new phone number:

4243653

Suucefully updated

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

5

Enter the ID to be deletd:

3

Successfully deleted

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

2

ID NAME AREA Ph Number

1 anu perinkary 87654321980

6 subina kannur 4243653

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

6

**PROGRAM NO: 29**

**AIM: Program to update record in a file**.

**ALGORITHM**

Step1: start

Step 2: create a structure emp

Step 3: create a structure variable e

Step 4: declare the file pointer fp

Step 5: open the file with w mode

Step 6: read number of employees number,name and salary

Step 7: close the file

Step 8: open the file with r mode

Step 9: if fp equal to null goto step 10 else goto step11

Step10: exit

Step 11: read the position to be updated

Step 12: write new data

Step 13: close the file

Step 14: open the file with r mode

Step 15:read the data from the file

Step 16:stop

**PROGRAM**

#include<stdio.h>

struct emp

{

int empno;

char name[20];

float salary;

};

int main(void)

{

struct emp e;

FILE \*fp;

int i,n,position,recno;

fp=fopen("emp.txt","w");

printf("Enter the number of employees\n");

scanf("%d",&n);

printf("Enter empno,name and salary %d employees\n",n);

for(i=1;i<=n;i++)

{

scanf("%d%s%f",&e.empno,e.name,&e.salary);

fwrite(&e,sizeof(e),1,fp);

}

fclose(fp);

printf("Contents of the file emp.txt\n");

fp=fopen("emp.txt","r");

while(fread(&e,sizeof(e),1,fp))

{

printf("%d\t%s\t%f\t\n",e.empno,e.name,e.salary);

}

fclose(fp);

fp=fopen("emp.txt","r+");

if(fp==NULL)

{

printf("emp.txt cannot be opened");

exit(0);

}

printf("Enter recno of the record to be updated\n");

scanf("%d",&recno);

fseek(fp,(recno-1)\*sizeof(e),SEEK\_SET);

printf("Enter new details of the employee\n");

scanf("%d%s%f",&e.empno,e.name,&e.salary);

fwrite(&e,sizeof(e),1,fp);

fclose(fp);

printf("Contents of the file emp.txt\n");

fp=fopen("emp.txt","r");

while(fread(&e,sizeof(e),1,fp))

{

printf("%d\t%s\t%f\t\n",e.empno,e.name,e.salary);

}

fclose(fp);

return 0;

}

**OUTPUT**

Enter the number of employees

4

Enter empno,name and salary 4 employees

123 Siyana 30000

124 Savina 40000

125 Mathew 45000

127 Hiran 30000

Contents of the file emp.txt

123 Siyana 30000.000000

124 Savina 40000.000000

125 Mathew 45000.000000

127 Hiran 30000.000000

Enter recno of the record to be updated

4

Enter new details of the employee

126 Kiran 30000

Contents of the file emp.txt

123 Siyana 30000.000000

124 Savina 40000.000000

125 Mathew 45000.000000

126 Kiran 30000.000000

**PROGRAM NO: 30**

**AIM: Command line argument.**

**ALGORITHM**

Step 1 : start

Step 2 : create a file pointer fp

Step 3 : print the value of argc

Step 4 : open a file with w mode

Step 5 : set i equal to 2

Step 6 : if i less than argc go to step 7 else go to step 9

Step 7 : print the value argv[i]

Step 8 : i equal to i + 1 go to step 6

Step 9 : close the file

Step 10 : print the value of argv[1]

Step 11 : open the file with r mode

Step 12 : set i equal to 2

Step 13 : if i less than argc go to step 14 else go to step 17

Step 14 : read the value of chfrom the file

Step 15 : print the value of ch

Step 16 : i equal to i + 1 go to step 13

Step 17 : close the file

Step 18 : stop

**PROGRAM**

#include<stdio.h>

int main(int argc,char \*argv[])

{

FILE \*fp;

char ch[20];

int i;

printf("Number of arguments %d\n",argc);

fp=fopen(argv[1],"w");

for(i=2;i<argc;i++)

fprintf(fp,"%s\n",argv[i]);

fclose(fp);

printf("Content of %s file \n",argv[1]);

fp=fopen(argv[1],"r");

for(i=2;i<argc;i++)

{

fscanf(fp,"%s",ch);

printf("%s",ch);

}

fclose(fp);

}

**OUTPUT:**

[Aleesha@DBCPC25 c record new]$ ./a.out sample.txt good morning

number of arguments 4

content of sample.txt file

goodmorning

**DATABASE MANAGEMENT SYSTEM**

**USING POSTGRESQL**

**DDL AND DML COMMANDS**

**1.Create a book table that contains book id,name and author of book.**

mcollege=# create table book(id int primary key,name varchar(20),author varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "book\_pkey" for table "book"

CREATE TABLE

**2. Insert rows into the table.**

mcacollege=# insert into book values(200,'yanthram','malayattor');

INSERT 0 1

mcacollege=# insert into book values(300,'adu jeevitham','benyamin');

INSERT 0 1

mcacollege=# insert into book values(400,'balyakalasagi','basheer');

INSERT 0 1

**3. Show the contents of the table.**

mcacollege=# select \* from book;

id | name | author

-----+------------------+-----------

200 | yanthram | malayattor

300 | adu jeevitham | benyamin

400 | balyakalasagi | basheer

(3 rows)

**4. Alter the book table contain price of a book.**

mcacollege=# alter table book add price int;

ALTER TABLE

**5. Show the contents of the table.**

mcacollege=# select \* from book;

id | name | author | price

-----+--------------+-----------+------

200 | yanthram | malayattor |

300 | adu jeevitham | benyamin |

400 | balyakalasagi | basheer |

(3 rows)

**6. Update the price of each book**

mcacollege=# update book set price=155 where id=200;

UPDATE 1

mcacollege=# update book set price=120 where id=300;

UPDATE 1

mcacollege=# update book set price=220 where id=400;

UPDATE 1

**7. Show the content of the table.**

mcacollege=# select \* from book;

id | name | author | price

-----+--------------+-----------+------

200 | yanthram | malayattor | 155

300 | adu jeevitham | benyamin | 120

400 | balyakalasagi | basheer | 220

(3 rows)

**8.Alter the table by deleting the column author.**

mcacollege=# alter table book drop author;

ALTER TABLE

**9.Show the contents of the table.**

mcacollege=# select \* from book;

id | name | price

-----+--------------+-----

200 | yanthram | 155

300 | adu jeevitham | 120

400 | balyakalasagi | 220

(3 rows)

**10. Delete some rows from book.**

mcacollege=# delete from book where id=300;

DELETE 1

**11. Show the content of the table.**

mcacollege=# select \* from book;

id | name | price

-----+--------------+------

200 | yanthram | 155

400 | balyakalasagi | 220

(2 rows)

**12. Drop the book table.**

mcacollege=# drop table book;

DROP TABLE

**INTEGRITY CONSTRAINTS**

**1. Create table department with fields department\_no number,name string,location number.**

mcacollege=# create table depp(dno int primary key, dname varchar(20),location int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "depp\_pkey" for table "depp"

CREATE TABLE

**2. Enter values for department table**

mcacollege=# insert into depp values(101,'m ca',1);

INSERT 0 1

mcacollege=# insert into depp values(102,'biology',1);

INSERT 0 1

mcacollege=# insert into depp values(103,'maths',2);

INSERT 0 1

mcacollege=# insert into depp values(104,'english',3);

INSERT 0 1

mcacollege=# insert into depp values(105,'chemistry',3);

INSERT 0 1

**3. Show the contents of department table.**

mcacollege=# select \* from depp;

dno | dname | location

-----+----------+---------

101 | mca | 1

102 | biology | 1

103 | maths | 2

104 | english | 3

105 | chemistry | 3

(5 rows)

**4. Create table employee with fields employe\_id number,name string,address string,salary number,dpartment\_no number.**

mcacollege=# create table employ(eid int primary key, ename varchar(20),address varchar(10),salary int,dno int references depp);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "employ\_pkey" for table "employ"

CREATE TABLE

**5. Insert rows into employee table.**

mcacollege=# insert into employ values(201,'amala','arumakel',10000,101);

INSERT 0 1

mcacollege=# insert into employ values(202,'sreeedar','valayil',16000,102);

INSERT 0 1

mcacollege=# insert into employ values(203,'sreeraj','anaswara',20000,102);

INSERT 0 1

mcacollege=# insert into employ values(204,'vinaya','mannil',18000,103);

INSERT 0 1

mcacollege=# insert into employ values(205,'arjun','vellara',18000,101);

INSERT 0 1

**6. Show the contents of employee table.**

mcacollege=# select \* from employ;

eid | ename | address | salary | dno

-----+---------+---------+-------+----

201 | amala | arumakel | 10000 | 101

202 | sreeedar | valayil | 16000 | 102

203 | sreeraj | anaswara | 20000 | 102

204 | vinaya | mannil | 18000 | 103

205 | arjun | vellara | 18000 | 101

(5 rows)

**7. Retrieve the total number of employees working in mca department.**

mcacollege=# select count(ename)from employ e ,depp d where d.dno=e.dno and dname='mca';

count

-------

2

(1 row)

**8. Retrieve name of employee and his department.**

mcacollege=# select e.ename,d.dname from depp d,employ e where d.dno=e.dno;

ename | dname

----------+--------

amala | mca

sreeedar | biology

sreeraj | biology

vinaya | maths

arjun | mca

(5 rows)

**9. Retrieve name of the employee who get the highest salary.**

mcacollege=# select ename,salary from employ where salary=(select max(salary) from employ);

ename | salary

---------+-------

sreeraj | 20000

(1 row)

**10. Retrieve name of the employee whose salary is between 10000 and 20000.**

mcacollege=# select ename from employ where salary between 10000 and 20000;

ename

----------

amala

sreeedar

sreeraj

vinaya

arjun

(5 rows)

**11. Retrieve name of the employee who get the highest salary from mca department.**

mcacollege=# select e.ename from employ e where salary=(select max(salary)from employ e,depp d where d.dno=e.dno and dname='mca');

ename

-------

arjun

(1 row)

**AGGREGATE FUNCTIONS**

**1.Create table foodcart with fields food id, food name, date and sold.**

mcacollege=# create table foodcart(fid int primary key,fname varchar(10),date date,sold int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "foodcart\_pkey" for table "foodcart"

CREATE TABLE

**2.Inert tuples into foodcart.**

mcacollege=# insert into foodcart values(100,'pizza','11/12/2009',30);

INSERT 0 1

mcacollege=# insert into foodcart values(101,'sandwitch','12/12/2009',80);

INSERT 0 1

mcacollege=# insert into foodcart values(103,'burger','12/12/2009',48);

INSERT 0 1

mcacollege=# insert into foodcart values(102,'cutlet','05/12/2009',79);

INSERT 0 1

mcacollege=# insert into foodcart values(104,'hotdog','06/12/2009',109);

INSERT 0 1

**3.Show the contents of foodcart.**

mcacollege=# select \* from foodcart;

fid | fname | date | sold

-----+-----------+---------------+------

100 | pizza | 2009-11-12 | 30

101 | sandwitch| 2009-12-12 | 80

103 | burger | 2009-12-12 | 48

102 | cutlet | 2009-05-12 | 79

104 | hotdog | 2009-06-12 | 109

(5 rows)

**4. Alter the table by adding field price.**

mcacollege=# alter table foodcart add price int;

ALTER TABLE

**5. Show the contents of foodcart.**

mcacollege=# select \* from foodcart;

fid | fname | date | sold | price

-----+-----------+----------------+------+-------

100 | pizza | 2009-11-12 | 30 |

101 | sandwitch | 2009-12-12 | 80 |

103 | burger | 2009-12-12 | 48 |

102 | cutlet | 2009-05-12 | 79 |

104 | hotdog | 2009-06-12 | 109|

(5 rows)

**6. Insert values to price field.**

mcacollege=# update foodcart set price=30 where fid=100;

UPDATE 1

mcacollege=# update foodcart set price=25 where fid=101;

UPDATE 1

mcacollege=# update foodcart set price=10 where fid=104;

UPDATE 1

mcacollege=# update foodcart set price=15 where fid=103;

UPDATE 1

mcacollege=# update foodcart set price=10 where fid=102;

UPDATE 1

**7. Show the contents of foodcart table.**

mcacollege=# select \* from foodcart;

fid | fname | date | sold | price

-----+-------------+------------+------+-------

100 | pizza | 2009-11-12 | 30 | 30

101 | sandwitch | 2009-12-12 | 80 | 25

104 | hotdog | 2009-06-12 | 109 | 10

103 | burger | 2009-12-12 | 48 | 15

102 | cutlet | 2009-05-12 | 79 | 10

(5 rows)

**8. Delete one row from foodcart.**

mcacollege=# delete from foodcart where fid=104;

DELETE 1

**9. Show the conents of foodcart.**

mcacollege=# select \* from foodcart;

fid | fname | date | sold | price

-----+-------------+--------------+------+-------

100 | pizza | 2009-11-12 | 30 | 30

101 | sandwitch | 2009-12-12 | 80 | 25

103 | burger | 2009-12-12 | 48 | 15

102 | cutlet | 2009-05-12 | 79 | 10

(4 rows)

**10. Retrieve number of items in foodcart.**

mcacollege=# select count(\*)from foodcart;

count

-------

4

(1 row)

**11. Retrieve sum of sold in food cart.**

mcacollege=# select sum(sold)from foodcart;

sum

-----

237

(1 row)

**12. Retrieve average price from foodcart.**

mcacollege=# select avg(price)from foodcart;

avg

---------------------

20.0000000000000000

(1 row)

**13. Retrieve maximum price from foodcart.**

mcacollege=# select max(price)from foodcart;

max

-----

30

(1 row)

**14. Retrieve minimum sold from foodcart.**

mcacollege=# select min(sold)from foodcart;

min

-----

30

(1 row)

**15. Insert values into foodcart.**

mcacollege=# insert into foodcart values(105,'pizza','06/12/2009',100,45);

INSERT 0 1

**16. Show the contents of foodcart.**

mcacollege=# select \* from foodcart;

fid | fname | date | sold | price

-----+-------------+--------------+------+-------

100 | pizza | 2009-11-12 | 30 | 30

101 | sandwitch | 2009-12-12 | 80 | 25

103 | burger | 2009-12-12 | 48 | 15

102 | cutlet | 2009-05-12 | 79 | 10

105 | pizza | 2009-06-12 | 100 | 45

(5 rows)

**17. Show the total sold of foodcart table group by food name.**

mcacollege=# select fname ,sum(sold)as totalsold from foodcart group by fname;

fname | totalsold

-------------+-----------

sandwitch | 80

burger | 48

pizza | 130

cutlet | 79

(4 rows)

**18. Retrieve food name and average price where average price is greater than 15.**

mcacollege=# select fname , avg(price) from foodcart group by fname having avg(price) >15;

fname | avg

-------------+---------------------

sandwitch | 25.0000000000000000

pizza | 37.5000000000000000

(2 rows)

**SET OPERATIONS**

**1. Create table depositor.**

dbms=# Create table depositor(id int primary key,name varchar(20),amount int,branch varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "depositor\_pkey" for table "depositor"

CREATE TABLE

**2. Insert values into depositor.**

dbms=# Insert into depositor values(101,'Ajay',30000,'kannur');

INSERT 0 1

dbms=# Insert into depositor values(102,'Sayooj',25000,'Iritty');

INSERT 0 1

dbms=# Insert into depositor values(103,'Sania',35000,'kannur');

INSERT 0 1

**3.Show the contents of depositor.**

dbms=# select \* from depositor;

id | name | amount | branch

-----+--------+--------+--------

101 | Ajay | 30000 | kannur

102 | Sayooj | 25000 | Iritty

103 | Sania | 35000 | kannur

(3 rows)

**4.Create table borrower.**

dbms=# Create table borrower(id int primary key,name varchar(20),amount int,branch varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "borrower\_pkey" for table "borrower"

CREATE TABLE

**5. Insert values into borrower.**

dbms=# Insert into borrower values(111,'Ajay',35000,'kannur');

INSERT 0 1

dbms=# Insert into borrower values(112,'Rahul',15000,'mattannur');

INSERT 0 1

dbms=# Insert into borrower values(113,'Ziya',15000,'iritty');

INSERT 0 1

**6. Show the contents of borrower.**

dbms=# select \* from borrower;

id | name | amount | branch

-----+-------+--------+-----------

111 | Ajay | 35000 | kannur

112 | Rahul | 15000 | mattannur

113 | Ziya | 15000 | iritty

(3 rows)

**7. Perform union operation with depositor and borrower table.**

dbms=# (select name from borrower)union(select name from depositor);

name

--------

Ziya

Sania

Rahul

Ajay

Sayooj

(5 rows)

**8. Perform intersect operation with depositor and borrower table.**

dbms=# (select name from borrower)intersect(select name from depositor);

name

------

Ajay

(1 row)

**9. Perform except** **operation with depositor and borrower table.**

dbms=# (select name from borrower)except(select name from depositor);

name

-------

Rahul

Ziya

(2 rows)

**VIEW**

**1. Create a table project with fields project\_id,project\_name,and hours.**

mcacollege=# create table project(pid int primary key,pname varchar(10),hours int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "project\_pkey" for table "project"

CREATE TABLE

**2. Insert tuples into project table.**

mcacollege=# insert into project values(1,'hms',7);

INSERT 0 1

mcacollege=# insert into project values(2,'cvms',9);

INSERT 0 1

mcacollege=# insert into project values(3,'facascs',13);

INSERT 0 1

mcacollege=# insert into project values(4,'library',11);

INSERT 0 1

mcacollege=# insert into project values(5,'officemgmt',16);

INSERT 0 1

**3. Show the contents of project.**

mcacollege=# select \* from project;

pid | pname | hours

-----+-----------+------

1 | hms | 7

2 | cvms | 9

3 | facascs | 13

4 | library | 11

5 | officemgmt | 16

(5 rows)

**4. Create a table employee**

mcacollege=# create table employy(eid int primary key,fname varchar(10),lname varchar(10),pid int references project);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "employy\_pkey" for table "employy"

CREATE TABLE

**5. Insert values into employee table.**

mcacollege=# insert into employy values(101,'anu','radha',1);

INSERT 0 1

mcacollege=# insert into employy values(103,'sona','jacob',3);

INSERT 0 1

mcacollege=# insert into employy values(102,'ebi','sakharia',3);

INSERT 0 1

mcacollege=# insert into employy values(104,'neena','roy',3);

INSER T 0 1

mcacollege=# insert into employy values(105,'zera','elizebath',5);

INSERT 0 1

**6. Show the contents of employee table.**

mcacollege=# select \* from employy;

eid | fname | lname | pid

-----+------+----------+----

101 | anu | radha | 1

103 | sona | jacob | 3

102 | ebi | sakharia | 3

104 | neena | roy | 3

105 | zera | elizebath | 5

(5 rows)

**7. Create a view.**

mcacollege=# create view projectemployy as select fname,lname,pname,hours from employy e,project p where e.pid=p.pid;

CREATE VIEW

**8. Show the contents of new view.**

mcacollege=# select \* from projectemployy;

fname | lname | pname | hours

-------+----------+-----------+------

anu | radha | hms | 7

sona | jacob | facascs | 13

ebi | sakharia | facascs | 13

neena | roy | facascs | 13

zera | elizebath | officemgmt | 16

(5 rows)

**9. Select value from view where project name is equal to ’facascs’**

college=# select \* from projectemployy where pname='facascs';

fname | lname | pname | hours

-------+---------+--------+------

sona | jacob | facascs | 13

ebi | sakharia | facascs | 13

neena | roy | facascs | 13

(3 rows)

**10. Select value from view .**

mcacollege=# select \* from projectemployy where hours < 13;

fname | lname | pname | hours

-------+------+------+------

anu | radha | hms | 7

(1 row)

**FUNCTION**

**I. Create a function to retrieve the mark of a student.**

**1. Create table student.**

mcacollege=# create table stud(id int primary key,name varchar(15),mark int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "stud\_pkey" for table "stud"

CREATE TABLE

**2.Insert values into student table.**

mcacollege=# insert into stud values(100,'anna',67);

INSERT 0 1

mcacollege=# insert into stud values(101,'manu',85);

INSERT 0 1

mcacollege=# insert into stud values(102,'roy',55);

INSERT 0 1

mcacollege=# insert into stud values(103,'ajin',65);

INSERT 0 1

mcacollege=# insert into stud values(104,'harsha',78);

INSERT 0 1

**3. Show the contents of student table.**

mcacollege=# select \* from stud;

id | name | mark

-----+--------+------

100 | anna | 67

101 | manu | 85

102 | roy | 55

103 | ajin | 65

104 | harsha | 78

(5 rows)

**4.** mcacollege=# create function plpgsql\_call\_handler() returns opaque as '$libdir/plpgsql.so' language'c';

CREATE FUNCTION

**5.** mcacollege=# create language 'plpgsql' handler plpgsql\_call\_handler lancompiler 'pl/pgsql';

NOTICE: using pg\_pltemplate information instead of CREATE LANGUAGE parameters

CREATE LANGUAGE

create function fun1(text) returns integer as'

declare

a int;

begin

select mark into a from stud where name=$1;

return a;

end

'language'plpgsql';

CREATE FUNCTION

**6. Show the contents of student table.**

mcacollege=# select \* from stud;

id | name | mark

-----+--------+------

100 | anna | 67

101 | manu | 85

102 | roy | 55

103 | ajin | 65

104 | harsha | 78

(5 rows)

mcacollege=# select fun1('anna');

fun1

------

67

(1 row)

mcacollege=# select fun1('manu');

fun1

------

85

(1 row)

mcacollege=# select fun1('ajin');

fun1

------

65

(1 row)

mcacollege=# select fun1('roy');

fun1

------

55

(1 row)

mcacollege=# select fun1('harsha');

fun1

------

78

(1 row)

**II. Create a function to add two numbers.**

create function sum(a int,b int)returns integer as'

begin

return a+b;

end

'language'plpgsql';

bash-4.1$ psql mcacollege

mcacollege=# \i funsum.sql

CREATE FUNCTION

mcacollege=# select sum(2,7);

sum

-----

9

(1 row)

mcacollege=# select sum(4,7);

sum

-----

11

(1 row)

**TRIGGER**

**1. Create table student.**

mcacollege=# create table stud(id int primary key,name varchar(15),mark int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "stud\_pkey" for table "stud"

CREATE TABLE

**2. Insert values into student.**

mcacollege=# insert into stud values(100,'anna',67);

INSERT 0 1

mcacollege=# insert into stud values(101,'manu',85);

INSERT 0 1

mcacollege=# insert into stud values(102,'roy',55);

INSERT 0 1

mcacollege=# insert into stud values(103,'ajin',65);

INSERT 0 1

mcacollege=# insert into stud values(104,'harsha',78);

INSERT 0 1

**3. Show the contents of student.**

mcacollege=# select \* from stud;

id | name | mark

-----+--------+------

100 | anna | 67

101 | manu | 85

102 | roy | 55

103 | ajin | 65

104 | harsha | 78

(5 rows)

**4. Create trigger.**

drop function fun2();

create function fun2() returns trigger as'

begin

update stud set mark=mark+30;

return new;

end

'language'plpgsql';

-bash-4.1$ psql mcacollege

CREATE FUNCTION

mcacollege=# create trigger trig after insert on stud execute procedure fun2();

CREATE TRIGGER

mcacollege=# insert into stud values(105,'meera',48);

INSERT 0 1

**5. Show the contents of student table.**

mcacollege=# select \* from stud;

id | name | mark

-----+---------+------

100 | anna | 97

101 | manu | 115

102 | roy | 85

103 | ajin | 95

104 | harsha | 108

105 | meera | 78

(6 rows)

**MATHEMATICAL FUNCTIONS**

**1.Sin()**

dbms=# SELECT sin(90);

sin

-------------------

0.893996663600558

(1 row)

**2. cos()**

dbms=# SELECT cos(0);

cos

-----

1

(1 row)

**3. tan()**

dbms=# SELECT tan(45);

tan

------------------

1.61977519054386

(1 row)

**4. Sqrt()**

dbms=# SELECT sqrt(25);

sqrt

------

5

(1 row)

**5. pow()**

dbms=# SELECT pow(2,5);

pow

-----

32

(1 row)

**6. pi()**

dbms=# SELECT pi();

pi

------------------

3.14159265358979

(1 row)

**7.radians()**

dbms=# SELECT radians(90);

radians

-----------------

1.5707963267949

(1 row)

**8. log()**

dbms=# SELECT log(180);

log

-----------------

2.25527250510331

(1 row)

**9.abs()**

dbms=# SELECT abs(-15);

abs

-----

15

(1 row)

**10. atan()**

dbms=# SELECT atan(30);

atan

------------------

1.53747533091665

(1 row)

**11. acos()**

dbms=# SELECT acos(1);

acos

------

0

(1 row)

**12.trunc()**

dbms=# SELECT trunc(18.23);

trunc

-------

18

(1 row)

**13.round()**

dbms=# SELECT round(45.43);

round

-------

45

(1 row)

**14.round()**

dbms=# SELECT round(45.435310,2);

round

-------

45.44

(1 row)

**15.random()**

dbms=# SELECT random();

random

------------------

0.13433201238513

(1 row)

**16.exp()**

dbms=# SELECT exp(20);

exp

-----------------

485165195.40979

(1 row)

**17.floor()**

dbms=# SELECT floor(1.01);

floor

-------

1

(1 row)

**18.cbrt()**

dbms=# SELECT cbrt(8);

cbrt

------

2

(1 row)

**19.ln()**

dbms=# SELECT ln(27) as natural\_log;

natural\_log

------------------

3.29583686600433

(1 row)

**20.ceil**

dbms=# SELECT ceil(2.7) ;

ceil

------

3

(1 row)

**CHARACTER STRING FUNCTIONS**

dbms=# SELECT ascii('T');

ascii

-------

84

(1 row)

dbms=# SELECT btrim('hello world','world');

btrim

--------

hello

(1 row)

dbms=# SELECT ltrim(' world');

ltrim

-------

world

(1 row)

dbms=# SELECT rtrim(' hello ');

rtrim

--------------

hello

(1 row)

dbms=# SELECT initcap('kiran');

initcap

---------

Kiran

(1 row)

dbms=# SELECT length('kiran');

length

--------

5

(1 row)

dbms=# SELECT substr('helloworld',4);

substr

---------

loworld

(1 row)

dbms=# SELECT substr('helloworld',6,5);

substr

--------

world

(1 row)

dbms=# SELECT lower('MATHEW');

lower

--------

mathew

(1 row)

dbms=# SELECT upper('Joseph');

upper

--------

JOSEPH

(1 row)

dbms=# SELECT chr(81);

chr

-----

Q

(1 row)

dbms=# SELECT translate('Good Morning','Morning','Evening');

translate

--------------

Good Evening

(1 row)

**TRANSACTION**

**1. Show the contents of library table.**

mcollege=# select \* from library;

lid | bookname | author | prize

-----+-----------------------------+---------------------+------

102 | bramanam | joycy | 110

103 | yanthram | malayatoor | 130

104 | arachar | k.r meera | 130

105 | nirmatalamputhakalam | kamala surayya | 230

106 | neelambari | kamala surayya | 210

107 | huckle berry fin | mark | 300

108 | tom soyar | mark | 200

200 | alchimist | paulo koylo | 240

101 | peythozhiyum neram | sara | 200

109 | olivar twist | chalse dikkance | 240

(10 rows)

mcacollege=# begin;

BEGIN

mcacollege=# update library set prize=145 where lid=102;

UPDATE 1

mcacollege=# update library set prize=145 where lid=103;

UPDATE 1

mcacollege=# delete from library where lid=104;

DELETE 1

mcacollege=# select \* from library;

lid | bookname | author | prize

-----+---------------------------- +----------------+------

105 | nirmatalamputhakalam | kamala surayya | 230

106 | neelambari | kamala surayya | 210

107 | huckle berry fin | mark | 300

108 | tom soyar | mark | 200

200 | alchimist | paulo koylo | 240

101 | peythozhiyum neram | sara | 200

109 | olivar twist | chalse dikkance | 240

102 | bramanam | joycy | 145

103 | yanthram | malayatoor | 145

(9 rows)

mcacollege=# commit;

COMMIT

mcacollege=# begin;

BEGIN

mcacollege=# alter table library add copy int;

ALTER TABLE

mcacollege=# update library set copy=145 where lid=102;

UPDATE 1

mcacollege=# update library set copy=26 where lid=106;

UPDATE 1

mcacollege=# select \* from library;

lid | bookname | author | prize | copy

-----+------------------------------+-------------------+--------+-----

105 | nirmatalamputhakalam | kamala surayya | 230 |

107 | huckle berry fin | mark | 300 |

108 | tom soyar | mark | 200 |

200 | alchimist | paulo koylo | 240 |

101 | peythozhiyum neram | sara | 200 |

109 | olivar twist | chalse dikkance | 240 |

103 | yanthram | malayatoor | 145 | 26

102 | bramanam | joycy | 145 | 145

106 | neelambari | kamala surayya | 210 | 26

(9 rows)

mcacollege=# rollback;

ROLLBACK

**CURSOR**

**1. Show the contents of student table.**

[mcacollege=# select \* from stud;

id | name | mark

-----+--------+-----

100 | anna | 127

101 | manu | 145

102 | roy | 115

103 | ajin | 125

104 | harsh | 138

105 | aleesha| 140

106 | asa | 130

107 | meera | 78

(8 rows)

mcacollege=# begin;

BEGIN

mcacollege=# declare st\_cur cursor for select name from stud;

DECLARE CURSOR

mcacollege=# fetch 5 from st\_cur;

name

--------

anna

manu

roy

ajin

harsha

(5 rows)

mcacollege=# fetch 3 from st\_cur;

name

---------

Aleesha

asa

meera(3 rows)

mcacollege=# move backward 5 in st\_cur;

MOVE 5

mcacollege=# fetch 3 from st\_cur;

name

---------

ajin

harsha

aleesha

(3 rows)

mcacollege=# fetch next from st\_cur;

name

------

asa

(1 row)

mcacollege=# fetch prior from st\_cur;

name

---------

aleesha

(1 row)

mcacollege=# move forward 5 in st\_cur;

MOVE 2

mcacollege=# fetch 0 from st\_cur;

name

------

(0 rows)

mcacollege=# move forward 5 in st\_cur;

MOVE 0

**INDEX**

mcacollege=# create index stud\_index on stud(name);

CREATE INDEX

mcacollege=# \d stud;

Table "public.stud"

Column | Type | Modifiers

--------+-----------------------+-----------

id | integer | not null

name | character varying(15) |

mark | integer |

Indexes:

"stud\_pkey" PRIMARY KEY, btree (id)

"st\_index" btree (name)

Trigger:

trig AFTER INSERT ON stud FOR EACH STATEMENT EXECUTE PROCEDURE fun2()

**ORDER PROCESSING DATABASE**

Consider the following relations for an order processing database application in a company. The primary keys are made bold and the data types are specified.

CUSTOMER( **custno**:int , cname:string , city:string )

ORDER( **orderno**:int , odate:date , custno:int , ord\_amt:int )

ORDER\_ITEM( **orderno**:int , **itemno**:int , quantity:int )

ITEM( **itemno**:int , unitprice:int )

SHIPMENT( **orderno**:int , **warehouseno**:int , ship\_date:date )

WAREHOUSE( warehouseno:int , city:string )

**1. Create the above tables by properly specifying the primary keys and foreign keys.**

dbms=# create table customer(custno int,cname varchar(10),city varchar(10),primary key(custno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "customer\_pkey" for table "customer"

CREATE TABLE

dbms=# create table order1(orderno int,odate date,custno int,ord\_amt int,primary key(orderno),foreign key(custno) references customer(custno));NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "order1\_pkey" for table "order1"

CREATE TABLE

dbms=# create table item(itemno int,unitprice int,primary key(itemno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "item\_pkey" for table "item"

CREATE TABLE

dbms=# create table order\_item(orderno int,itemno int,quantity int,primary key(orderno,itemno),foreign key(orderno) references order1(orderno),foreign key(itemno) references item(itemno) on delete cascade);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "order\_item\_pkey" for table "order\_item"

CREATE TABLE

dbms=# create table warehouse(warehouseno int,city varchar(10),primary key(warehouseno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "warehouse\_pkey" for table "warehouse"

CREATE TABLE

dbms=# create table shipment(orderno int,warehouseno int,ship\_date date,primary key(orderno,warehouseno),foreign key(orderno) references order1(orderno),foreign key(warehouseno) references warehouse(warehouseno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "shipment\_pkey" for table "shipment"

CREATE TABLE

**2. Enter at least five tuples for each relation and show the contents of each relation.**

dbms=# insert into customer values(100,'arnav','bangalore');

INSERT 0 1

dbms=# insert into customer values(101,'amitha','kochi');

INSERT 0 1

dbms=# insert into customer values(102,'babitha','kochi');

INSERT 0 1

dbms=# insert into customer values(103,'praveen','calicut');

INSERT 0 1

dbms=# insert into customer values(104,'ambika','trichur');

INSERT 0 1

dbms=# select \* from customer;

custno | cname | city

--------+--------+----------

100 | arnav | bangalore

101 | amitha | kochi

102 | babitha | kochi

103 | praveen | calicut

104 | ambika | trichur

(5 rows)

dbms=# insert into order1 values(200,'03-15-2017',101,35000);

INSERT 0 1

dbms=# insert into order1 values(201,'01-12-2016',100,53000);

INSERT 0 1

dbms=# insert into order1 values(202,'03-12-2016',104,5600);

INSERT 0 1

dbms=# insert into order1 values(203,'12-12-2016',103,45000);

INSERT 0 1

dbms=# insert into order1 values(204,'12-09-2016',102,12000);

INSERT 0 1

dbms=# select \* from order1;

orderno | odate | custno | ord\_amt

---------+-----------+-------+--------

200 | 2017-03-15 | 101 | 35000

201 | 2016-01-12 | 100 | 53000

202 | 2016-03-12 | 104 | 5600

203 | 2016-12-12 | 103 | 45000

204 | 2016-12-09 | 102 | 12000

(5 rows)

dbms=# insert into item values(155,38000);

INSERT 0 1

dbms=# insert into item values(156,5600);

INSERT 0 1

dbms=# insert into item values(158,12000);

INSERT 0 1

dbms=# insert into item values(200,15000);

INSERT 0 1

dbms=# insert into item values(202,22000);

INSERT 0 1

dbms=# select \* from item;

itemno | unitprice

--------+----------

155 | 38000

156 | 5600

158 | 12000

200 | 15000

202 | 22000

(5 rows)

dbms=# insert into order\_item values(200,156,1200);

INSERT 0 1

dbms=# insert into order\_item values(201,155,100);

INSERT 0 1

dbms=# insert into order\_item values(202,158,150);

INSERT 0 1

dbms=# insert into order\_item values(203,200,1500);

INSERT 0 1

dbms=# insert into order\_item values(204,202,250);

INSERT 0 1

dbms=# select \* from order\_item;

orderno | itemno | quantity

---------+-------+---------

200 | 156 | 1200

201 | 155 | 100

202 | 158 | 150

203 | 200 | 1500

204 | 202 | 250

(5 rows)

dbms=# insert into warehouse values(201,'calicut');

INSERT 0 1

dbms=# insert into warehouse values(202,'trivandrum');

INSERT 0 1

dbms=# insert into warehouse values(203,'kochi');

INSERT 0 1

dbms=# insert into warehouse values(204,'kochi');

INSERT 0 1

dbms=# insert into warehouse values(205,'kannur');

INSERT 0 1

dbms=# select \* from warehouse;

warehouseno | city

-------------+-----------

201 | calicut

202 | trivandrum

203 | kochi

204 | kochi

205 | kannur

(5 rows)

dbms=# insert into shipment values(200,201,'03-23-2017');

INSERT 0 1

dbms=# insert into shipment values(201,202,'05-13-2017');

INSERT 0 1

dbms=# insert into shipment values(202,203,'05-12-2016');

INSERT 0 1

dbms=# insert into shipment values(203,203,'12-12-2016');

INSERT 0 1

dbms=# insert into shipment values(203,204,'12-12-2016');

INSERT 0 1

dbms=# select \* from shipment;

orderno | warehouseno | ship\_date

---------+------------+-----------

200 | 201 | 2017-03-23

201 | 202 | 2017-05-13

202 | 203 | 2016-05-12

203 | 203 | 2016-12-12

203 | 204 | 2016-12-12

(5 rows)

**3. Produce a listing: custname , No\_of\_orders , Avg\_order\_amount , where the middle column is the total number of orders by the customer and the last column is the average order amount for that customer.**

dbms=# select c.custno,count(\*) as no\_of\_orders,avg(o.ord\_amt) as avg\_order\_amt from customer c,order1 o where c.custno=o.custno group by c.custno;

custno | no\_of\_orders | avg\_order\_amt

--------+-------------+-----------------------

101 | 1 | 35000.000000000000

104 | 1 | 5600.0000000000000000

102 | 1 | 12000.0000000000000000

100 | 1 | 53000.000000000000

103 | 1 | 45000.000000000000

(5 rows)

dbms=# delete from item where itemno=158;

DELETE 1

dbms=# select \* from item;

itemno | unitprice

--------+----------

155 | 38000

156 | 5600

200 | 15000

202 | 22000

(4 rows)

dbms=# select s.orderno from shipment s,warehouse w where s.warehouseno=w.warehouseno and w.city='kochi';

orderno

---------

202

203

203

(3 rows)

**EMPLOYEE DEPARTMENT RELATION**

Consider the database given below. The primary keys are made bold and the data types are specified.

emp (**empno** number, ename varchar, job varchar, deptno number, sal number)

dept (**deptno** number, dname varchar, loc varchar)

**1. Create the above tables by properly specifying the primary keys and foreign keys.**

mcacollege=# create table empl(eno int not null primary key,ename varchar(15),job varchar(15),dno int references dep,salary int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "empl\_pkey" for table "empl"

CREATE TABLE

mccollege=# create table dep(dno int not null primary key,dname varchar(15),location varchar(15));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "dep\_pkey" for table "dep"

CREATE TABLE

**2. Enter at least five tuples for each relation.**

mcacollege=# insert into dep values(10,'msw','1st floor');

INSERT 0 1

mcacollege=# insert into dep values(11,'bca','2nd floor');

INSERT 0 1

mcacollege=# insert into dep values(12,'ma','1st floor');

INSERT 0 1

mcacollege=# insert into dep values(13,'mcom','3rd floor');

INSERT 0 1

mcacollege=# insert into dep values(14,'bsw','3rd floor');

INSERT 0 1

mcacollege=# insert into empl values(10,'jijo','manager',10,27500);

INSERT 0 1

mcacollege=# insert into empl values(12,'roy','clerk',10,3500);

INSERT 0 1

mcacollege=# insert into empl values(13,'heera','accountant',14,13500);

INSERT 0 1

mcacollege=# insert into empl values(14,'Cristy','treasurer',13,15000);

INSERT 0 1

mcacollege=# insert into empl values(11,'anna','manager',10,37500);

INSERT 0 1

**3. Show the the contents of each relation.**

mcacollege=# select \* from dep;

dno | dname | location

-----+------+----------

10 | msw | 1st floor

11 | bca | 2nd floor

12 | ma | 1st floor

13 | mcom | 3rd floor

14 | bsw | 3rd floor

(5 rows)

mcacollege=# select \* from empl;

eno | ename | job | dno | salary

-----+-------+-------------+----+-------

10 | jijo | manager | 10 | 27500

12 | roy | clerk | 10 | 3500

13 | heera | accountant| 14 | 13500

14 | Cristy | treasurer | 13 | 15000

11 | anna | manager | 10 | 37500

(5 rows)

**4. Retrieve name of the employee who get the highest salary from msw department.**

mcacollege=#mcacollege=# select e.ename from empl e where salary=(select max(salary)from empl e,dep d where d.dno=e.dno and dname='msw');

ename

-------

anna

(1 row)

**5. List the records in the emp table order by salary in ascending order.**

mcacollege=# select \* from empl order by salary asc;

eno | ename | job | dno | salary

-----+--------+-------------+----+-------

12 | roy | clerk | 10 | 3500

13 | heera | accountant | 14 | 13500

14 | Cristy | treasurer | 13 | 15000

10 | jijo | manager | 10 | 27500

11 | anna | manager | 10 | 37500

(5 rows)

**6. Display deptno from the table employee avoiding the duplicated values.**

mcacollege=# select distinct dno from empl;

dno

-----

13

14

10

(3 rows)

**7. List the names of employees along with the department name.**

mcacollege=# select ename,dname from empl e,dep d where e.dno=d.dno;

ename | dname

----------+------

jijo | msw

roy | msw

heera | bsw

Cristy | mcom

anna | msw

(5 rows)

**INSTRUCTOR-DEPARTMENT RELATION**

**1. Create department relation.**

sample=# CREATE TABLE dep(dep\_name varchar(20) primary key,HOD varchar(20),location varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "dep\_pkey" for table "dep"

CREATE TABLE

**2. Insert tuples into relation.**

sample=# INSERT INTO dep values('cs','Anitha Haridhas','kannur');

INSERT 0 1

sample=# INSERT INTO dep values('English','Layra','chala');

INSERT 0 1

sample=# INSERT INTO dep values('Biothechnology','Nayana','kuthuparamb');

INSERT 0 1

sample=# INSERT INTO dep values('Maths','Aswathi','kuthuparamb');

INSERT 0 1

sample=# INSERT INTO dep values('Malayalam','Jessy','mattannur');

INSERT 0 1

**3. Show the contents of department relation.**

sample=# SELECT \* from dep;

dep\_name | hod | location

---------------------+--------------------+------------

cs | Anitha Haridhas | kannur

English | Layra | chala

Biothechnology | Nayana | kuthuparamb

Maths | Aswathi | kuthuparamb

Malayalam | Jessy | mattannur

(5 rows)

**4. Create an instructor relation.**

sample=# CREATE TABLE instruct(instrct\_id int primary key,name varchar(20),dep\_name varchar(20),foreign key(dep\_name) references dep,salary numeric(8,2),address varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "instruct\_pkey" for table "instruct"

CREATE TABLE

**5. Insert tuples into instructor relation.**

sample=# INSERT INTO instruct values (101,'subina','cs',20000,'kannur');

INSERT 0 1

sample=# INSERT INTO instruct values (102,'NIdhin','English',17000,'kannur');

INSERT 0 1

sample=# INSERT INTO instruct values (103,'Aswathi','Biothechnology',30000,'koothuparamb');

INSERT 0 1

sample=# INSERT INTO instruct values (104,'Anjali','Maths',25000,'mattannur');

INSERT 0 1

sample=# INSERT INTO instruct values (105,'Reshma','Malayalam',23000,'mattannur');

INSERT 0 1

**6. Show the content of instructor table.**

sample=# SELECT \* from instruct;

instrct\_id | name | dep\_name | salary | address

------------+--------+---------------+---------+-------------

101 | subina | cs | 20000.00 | kannur

102 | NIdhin | English | 17000.00 | kannur

103 | Aswathi | Biothechnology| 30000.00| koothuparamb

104 | Anjali | Maths | 25000.00 | mattannur

105 | Reshma | Malayalam | 23000.00 | mattannur

(5 rows)

**7. Retrieve name and department name from instructor table.**

sample=# SELECT name,dep\_name from instruct;

name | dep\_name

-------------+---------------

subina | cs

NIdhin | English

Aswathi | Biothechnology

Anjali | Maths

Reshma | Malayalam

(5 rows)

**8. Retrieve name and department name from salary where salary greater than 20000.**

sample=# SELECT name,dep\_name from instruct where salary>20000.00 ;

name | dep\_name

-------------+---------------

Aswathi | Biothechnology

Anjali | Maths

Reshma | Malayalam

(3 rows)

**9. Retrieve average salary from instructor.**

sample=# SELECT avg(salary) from instruct;

avg

--------------------

23000.000000000000

(1 row)

**10. Select average salary from instructor and group by department name.**

sample=# SELECT avg(salary) as avg\_salary from instruct group by dep\_name;

avg\_salary

------------------------

25000.000000000000

23000.000000000000

17000.0000000000000000

30000.000000000000

20000.000000000000

(5 rows)

**11. Select name from instructor where salary between 10000 and 20000.**

sample=# SELECT name from instruct where salary between 10000 and 20000;

name

--------

subina

NIdhin

(2 rows)

**12. Delete one row from instructor where department name equal to ‘cs’.**

sample=# DELETE from instruct where dep\_name='cs';

DELETE 1

**13. Show the contents of instructor table.**

sample=# SELECT \* from instruct;

instrct\_id | name | dep\_name | salary | address

------------+--------+---------------+---------+-------------

102 | NIdhin | English | 17000.00 | kannur

103 | Aswathi | Biothechnology | 30000.00 | koothuparamb

104 | Anjali | Maths | 25000.00 | mattannur

105 | Reshma | Malayalam | 23000.00 | mattannur

(4 rows)

**14. Update instructor salary.**

sample=# UPDATE instruct set salary=salary\*1.05;

UPDATE 4

sample=# SELECT \* from instruct;

instrct\_id | name | dep\_name | salary | address

------------+--------+---------------+---------+-------------

102 | NIdhin | English | 17850.00 | kannur

103 | Aswathi | Biothechnology | 31500.00 | koothuparamb

104 | Anjali | Maths | 26250.00 | mattannur

105 | Reshma | Malayalam | 24150.00 | mattannur

(4 rows)

**INSURANCE DATABASE**

**Consider the insurance database given below. The primary keys are made bold  and the data types are specified.**PERSON( **driver\_id**:string , name:string , address:string )  
CAR( **regno**:string , model:string , year:int )  
ACCIDENT( **report\_number**:int , accd\_date:date , location:string )  
OWNS( **driver\_id**:string , **regno**:string )  
PARTICIPATED( **driver\_id**:string , **regno**:string , **report\_number**:int , damage\_amount:int)

1**. Create the above tables by properly specifying the primary keys and foreign keys.**

bank=# create table person(driver\_id varchar(10) primary key,name varchar(20),address varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "person\_pkey" for table "person"

CREATE TABLE

bank=# create table car (regno varchar(10) primary key,model varchar(10),year int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "car\_pkey" for table "car"

CREATE TABLE

bank=# create table accident(report\_number int primary key,acc\_date date,location varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "accident\_pkey" for table "accident"

CREATE TABLE

bank=# create table owns( driver\_id varchar(10) references car,regno varchar(10)references car);

CREATE TABLE

bank=# create table participated(driver\_id varchar(10) references person,regno varchar(10) references car,report\_number int references accident);

CREATE TABLE

**2. Enter tuples for each relation and show the contents**

bank=#insert into person values('KL13','Nidhin','vallithodu');

bank=#insert into person values('KL14','varun','peratta');

bank=#insert into person values('KL15','arun','iritty');

bank=#insert into person values('KL16','arjun','perinkary');

bank=#insert into person values('KL17','vivek','peravoor');

bank=# select \* from person;

driver\_id | name | address

-----------+-------+-----------

KL13 | Nidhin | vallithodu

KL14 | varun | peratta

KL15 | arun | iritty

KL16 | arjun | perinkary

KL17 | vivek | peravoor

(5 rows)

bank=#insert into car values('KR100','BMW',2000);

bank=#insert into car values('KR101','audi',2015);

bank=#insert into car values('KR102','benz',2014);

bank=#insert into car values('KR103','maruthi',2001);

bank=#insert into car values('KR104','BMW',2002);

bank=# select \* from car;

regno | model | year

-------+--------+-----

KR100 | BMW | 2000

KR101 | audi | 2015

KR102 | benz | 2014

KR103 | maruthi | 2001

KR104 | BMW | 2002

(5 rows)

bank=#insert into accident values(455,'2/4/2015','kannur');

bank=#insert into accident values(456,'4/7/2008','kannur');

bank=#insert into accident values(457,'5/7/2008','kannur');

bank=#insert into accident values(458,'5/9/2009','thalasery');

bank=#insert into accident values(459,'6/9/2009','iritty');

bank=# select \* from accident;

report\_number | acc\_date | location

---------------+-----------+-----------

455 | 2015-02-04 | kannur

456 | 2008-04-07 | kannur

457 | 2008-05-07 | kannur

458 | 2009-05-09 | thalassery

459 | 2009-06-09 | iritty

(5 rows)

bank=#insert into owns values('KL13','KR100');

bank=#insert into owns values('KL14','KR101');

bank=#insert into owns values('KL15','KR102');

bank=#insert into owns values('KL16','KR103');

bank=#insert into owns values('KL17','KR104');

bank=# select \* from owns;

driver\_id | regno

-----------+------

KL13 | KR100

KL14 | KR101

KL15 | KR102

KL16 | KR103

KL17 | KR104

(5 rows)

bank=#insert into participated values('KL13','KR100',455,50000);

bank=#insert into participated values('KL14','KR101',456,100000);

bank=#insert into participated values('KL15','KR102',457,365420);

bank=#insert into participated values('KL16','KR103',458,30000);

bank=#insert into participated values('KL17','KR104',459,60000);

bank=# select \* from participated;

driver\_id | regno | report\_number | damage\_amount

-----------+------+--------------+--------------

KL13 | KR100 | 455 | 50000

KL14 | KR101 | 456 | 100000

KL15 | KR102 | 457 | 365420

KL16 | KR103 | 458 | 30000

KL17 | KR104 | 459 | 60000

(5 rows)

**3. Update the damage amount for the carwith specific regno in the accident with report number 456 to 250000**

bank=# update participated set damage\_amount=250000 where report\_number=456 and regno='KR101';

UPDATE 1

**4. Show the contents of participated table.**

bank=# select \* from participated;

driver\_id | regno | report\_number | damage\_amount

-----------+------+--------------+--------------

KL13 | KR100 | 455 | 50000

KL15 | KR102 | 457 | 365420

KL16 | KR103 | 458 | 30000

KL17 | KR104 | 459 | 60000

KL14 | KR101 | 456 | 250000

**5. Find the number of accidents in which cars belonging to a specific model were involved.**

bank=# select count(\*) as Totalcars from car c,participated p where c.regno=p.regno and c.model='BMW';

totalcars

-----------

2

(1 row)

**6. Find the total number of people who owned cars that were involved in accidents in the year 2015.**

bank=# select count(driver\_id) as people from owns o,car c where c.year=2015 and c.regno=o.regno;

people

--------

1

(1 row)

**BANKING ENTERPRISE**

**Consider the following database for a banking enterprise.**BRANCH( **branch\_name**:string , branch\_city:string , assets:real )  
ACCOUNT( **accno**:int , branch\_name:string , balance:real )  
DEPOSITOR( **customer\_name**:string , **accno**:int )  
CUSTOMER( **customer\_name**:string , customer\_street:string , customer\_city:string )  
LOAN( **loan\_number**:int , branch\_name:string , amount:real )  
BORROWER( **customer\_name**:string , **loan\_number**:int )

**1. Create the above tables by properly specifying the primary keys and foreign keys.**lab-dbms=# create table branch(branch\_name varchar(20) primary key,branch\_city varchar(20),assets numeric(8,2));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "branch\_pkey" for table "branch"

CREATE TABLE

lab-dbms=# create table account(acc\_no int primary key, branch\_name varchar(20) references branch,balance numeric(8,2));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "account\_pkey" for table "account"

CREATE TABLE

lab-dbms=# create table customer(customer\_name varchar(20) primary key,customer\_street varchar(20),customer\_city varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "customer\_pkey" for table "customer"

CREATE TABLE

lab-dbms=# create table depositor(acc\_no int references account,customer\_name varchar(20) references customer);

CREATE TABLE

lab-dbms=# create table loan(loan\_no int primary key,branch\_name varchar(20) references branch,account numeric(8,2));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "loan\_pkey" for table "loan"

CREATE TABLE

lab-dbms=# create table borrower(customer\_name varchar(20) references customer,loan\_no int references loan);

CREATE TABLE

**2. Enter at least five tuples for each relation**

lab-dbms=# insert into branch values('kannur','kerala',40000);

INSERT 0 1

lab-dbms=# insert into branch values('kochi','kerala',90000);

INSERT 0 1

lab-dbms=# insert into branch values('bangalore','karnataka',100000);

INSERT 0 1

lab-dbms=# insert into branch values('hyderbad','andrapredhesh',80000);

INSERT 0 1

lab-dbms=# insert into branch values('alapuzha','kerala',35000);

INSERT 0 1

lab-dbms=# insert into account values(101,'kannur',15000);

INSERT 0 1

lab-dbms=# insert into account values(102,'kochi',45000);

INSERT 0 1

lab-dbms=# insert into account values(103,'bangalore',55000);

INSERT 0 1

lab-dbms=# insert into account values(104,'hyderbad',35000);

INSERT 0 1

lab-dbms=# insert into account values(105,'alapuzha',22000);

INSERT 0 1

lab-dbms=# insert into customer values('Nidhin','kannur','kerala');

INSERT 0 1

lab-dbms=# insert into customer values('Sayooj','kochi','kerala');

INSERT 0 1

lab-dbms=# insert into customer values('Ramess','bangalore','karnataka');

INSERT 0 1

lab-dbms=# insert into customer values('Jaison','hyderbad','andrapredhesh');

INSERT 0 1

lab-dbms=# insert into customer values('allan','alapuzha','kerala');

INSERT 0 1

lab-dbms=# insert into depositor values(101,’Nidhin’);

INSERT 0 1

lab-dbms=# insert into depositor values(102,’Sayooj’);

INSERT 0 1

lab-dbms=# insert into depositor values(104,’Jaison’);

INSERT 0 1

lab-dbms=# insert into depositor values(105,’allen’);

INSERT 0 1

lab-dbms=# insert into loan values(1001,’kannur’,10000.00);

INSERT 0 1

lab-dbms=# insert into loan values(1002,’kochi’,30000.00);

INSERT 0 1

lab-dbms=# insert into loanr values(1003,’ bangalore’,50000.00);

INSERT 0 1

lab-dbms=# insert into loan values(1004,’ hyderbad’,80000.00);

INSERT 0 1

lab-dbms=# insert into loan values(1005,’alapuzha’,15000.00);

INSERT 0 1

lab-dbms=# insert into borrower values(’Nidhin’,1001);

INSERT 0 1

lab-dbms=# insert into borrower values(’Sayooj’,1002);

INSERT 0 1

lab-dbms=# insert into borrower values(’Ramees’,1003);

INSERT 0 1

lab-dbms=# insert into borrower values(’Jaison’,1004);

INSERT 0 1

lab-dbms=# insert into borrower values(’allen’,1005);

INSERT 0 1

**3. Show the contents of each table.**

lab-dbms=# select \* from branch;

branch\_name | branch\_city | assets

-------------+---------------+-----------

kannur | kerala | 40000.00

kochi | kerala | 90000.00

bangalore | karnataka | 100000.00

hyderbad | andrapredhesh | 80000.00

alapuzha | kerala | 35000.00

(5 rows)

lab-dbms=# select \* from account;

acc\_no | branch\_name | balance

--------+-------------+----------

101 | kannur | 15000.00

102 | kochi | 45000.00

104 | hyderbad | 35000.00

105 | alapuzha | 22000.00

(4 rows)

lab-dbms=# select \* from depositor;

acc\_no | customer\_name

--------+---------------

101 | Nidhin

102 | Sayooj

104 | Jaison

105 | allan

(4 rows)

lab-dbms=# select \* from loan;

loan\_no | branch\_name | account

---------+-------------+----------

1001 | kannur | 10000.00

1002 | kochi | 30000.00

1003 | bangalore | 50000.00

1004 | hyderbad | 80000.00

1005 | alapuzha | 15000.00

(5 rows)

lab-dbms=# select \* from borrower;

customer\_name | loan\_no

---------------+---------

Nidhin | 1001

Sayooj | 1002

Ramess | 1003

Jaison | 1004

allan | 1005

(5 rows)

lab-dbms=# select \* from customer;

customer\_name | customer\_street | customer\_city

---------------+-----------------+---------------

Nidhin | kannur | kerala

Sayooj | kochi | kerala

Ramess | bangalore | karnataka

Jaison | hyderbad | andrapredhesh

allan | alapuzha | kerala

(5 rows)

**4. Find all the customers who have an account at *all* the branches located in a specific city..**

lab-dbms=# select d.customer\_name from depositor d,account a where a.acc\_no=d.acc\_no and a.branch\_name='bangalore' group by d.customer\_name having count(\*)>=1;

customer\_name

---------------

Ramess

(1 row)

**5. Demonstrate how you delete all account tuples at every branch located in a specific city.**

lab-dbms=# delete from account where acc\_no in(select a.acc\_no from account a, branch b where a.branch\_name=b.branch\_name and b.branch\_city='karnataka');

DELETE 1

lab-dbms=# select \* from account;

acc\_no | branch\_name | balance

--------+------------+---------

101 | kannur | 15000.00

102 | kochi | 45000.00

104 | hyderbad | 35000.00

105 | alapuzha | 22000.00

(4 rows)

**STUDENT ENROLLMENT**

**Consider the following database of student enrollment in courses and books adopted for that course.**  
STUDENT( **regno**:string , name:string , major:string , bdate:date )  
COURSE**( courseno**:int , cname:string , dept:string )  
ENROLL**( regno**:string , **courseno**:int , **sem**:int , marks:int )  
BOOK\_ADOPTION( **courseno**:int , **sem**:int , book\_isbn:int )  
TEXT( **book\_isbn**:int , book\_title:string , publisher:string , author:string )

**1. Create the above tables by properly specifying the primary keys and foreign keys.**

dbms=# Create table students(regno varchar(20),bame varchar(20),major varchar(20),bdate date,primary key(regno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "students\_pkey" for table "students"

CREATE TABLE

bms=# Create table course(courseno int,cname varchar(10),dept varchar(10),primary key(courseno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "course\_pkey" for table "course"

CREATE TABLE

dbms=# create table enroll(regno varchar(10),courseno int,sem int,marks int,primary key(regno,courseno,sem),foreign key(regno)references students(regno),foreign key(courseno)references course(courseno));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "enroll\_pkey" for table "enroll"

CREATE TABLE

dbms=# create table text(book\_isbn int,book\_title varchar(10),publisher varchar(10),author varchar(10),primary key(book\_isbn));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "text\_pkey" for table "text"

CREATE TABLE

dbms=# create table book\_adoption(courseno int,sem int,book\_isbn int,primary key(courseno,sem),foreign key(courseno)references course(courseno),foreign key(book\_isbn)references text(book\_isbn));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "book\_adoption\_pkey" for table "book\_adoption"

CREATE TABLE

**2. Enter atleast five tuples for each relation.**

dbms=# insert into students values('dbc101','Amal','major','12/11/1995');

INSERT 0 1

dbms=# insert into students values('dbc102','Vinay','major','02/10/1995');

INSERT 0 1

dbms=# insert into students values('dbc103','Afna','major','01/03/1995');

INSERT 0 1

dbms=# insert into students values('dbc104','Siya','major','12/27/1995');

INSERT 0 1

dbms=# insert into students values('dbc105','Edison','major','05/09/1995');

INSERT 0 1

bms=# insert into course values(211,'BCA','CS');

INSERT 0 1

dbms=# insert into course values(210,'BCOM','COMMERCE');

INSERT 0 1

dbms=# insert into course values(212,'BSc.CS','CS');

INSERT 0 1

dbms=# insert into course values(213,'BSc.Maths','MATHS');

INSERT 0 1

dbms=# insert into course values(214,'BCA','CS');

INSERT 0 1

dbms=# insert into enroll values('dbc101',211,2,40);

INSERT 0 1

dbms=# insert into enroll values('dbc102',210,1,39);

INSERT 0 1

dbms=# insert into enroll values('dbc103',212,2,41);

INSERT 0 1

dbms=# insert into enroll values('dbc104',212,2,40);

INSERT 0 1

dbms=# insert into enroll values('dbc105',214,2,28);

INSERT 0 1

dbms=# insert into text values(124,'Cpp ','DD','Yaswant');

INSERT 0 1

dbms=# insert into text values(453,'C','DD','Yaswant');

INSERT 0 1

dbms=# insert into text values(765,'QA','DD','Agarwal');

INSERT 0 1

dbms=# insert into text values(565,'managment','Fary','Gupta');

INSERT 0 1

dbms=# insert into text values(983,'CG','DC','Albert');

INSERT 0 1

dbms=# insert into book\_adoption values(211,1,453);

INSERT 0 1

dbms=# insert into book\_adoption values(212,2,124);

INSERT 0 1

dbms=# insert into book\_adoption values(214,1,565);

INSERT 0 1

dbms=# insert into book\_adoption values(212,1,765);

INSERT 0 1

dbms=# insert into book\_adoption values(210,2,565);

INSERT 0 1

**3. Show the contents of each table.**

dbms=# select \* from students;

regno | bame | major | bdate

--------+--------+-------+------------

dbc101 | Amal | major | 1995-12-11

dbc102 | Vinay | major | 1995-02-10

dbc103 | Afna | major | 1995-01-03

dbc104 | Siya | major | 1995-12-27

dbc105 | Edison | major | 1995-05-09

(5 rows)

dbms=# select \* from enroll;

regno | courseno | sem | marks

--------+----------+-----+-------

dbc101 | 211 | 2 | 40

dbc102 | 210 | 1 | 39

dbc103 | 212 | 2 | 41

dbc104 | 212 | 2 | 40

dbc105 | 214 | 2 | 28

(5 rows)

dbms=# select \* from text;

book\_isbn | book\_title | publisher | author

-----------+------------+-----------+---------

124 | Cpp | DD | Yaswant

453 | C | DD | Yaswant

765 | QA | DD | Agarwal

565 | managment | Fary | Gupta

983 | CG | DC | Albert

(5 rows)

dbms=# select \* from book\_adoption;

courseno | sem | book\_isbn

----------+-----+-----------

211 | 1 | 453

212 | 2 | 124

214 | 1 | 565

212 | 1 | 765

210 | 2 | 565

(5 rows)

dbms=# select \* from course;

courseno | cname | dept

----------+-----------+----------

210 | BCOM | COMMERCE

212 | BSc.CS | CS

213 | BSc.Maths | MATHS

214 | BCA | CS

211 | BCA | CA

(5 rows)

**4. List any department that has *all* its books published by a specific publisher**

dbms=# select dept from course c,book\_adoption b,text t where c.courseno=b.courseno and t.publisher='DD' and t.book\_isbn=b.book\_isbn ;

dept

------

CA

CS

CS

(3 rows)

**5. Produce a list of text books ( includes courseno , book\_isbn , book\_title ) in the alphabetical order for courses offered by the 'CS' department that use more than two books.**

dbms=# select c.courseno,b.book\_isbn,t.book\_title from course c,book\_adoption b,text t where c.courseno=b.courseno and t.book\_isbn=b.book\_isbn and c.dept='CS' order by courseno ;

courseno | book\_isbn | book\_title

----------+-----------+------------

212 | 124 | Cpp

212 | 765 | QA

214 | 565 | managment

(3 rows)

**LINUX OPERATING SYSTEM**

**USER MANAGEMANT COMMANDS**

**COMMAND**

useradd - Create a new user or update default new user information.

**SYNTAX**

useradd [-c comment] [-d home\_dir]

[-e expire\_date] [-f inactive\_time]

[-g initial\_group] [-G group[,...]]

[-m [-k skeleton\_dir] | -M] [-n] [-o] [-p passwd] [-r]

[-s shell] [-u uid] login

**DESCRIPTION**

**Creating New Users**

When invoked without the -D option, the useradd command creates a new user account using the values specified on the command line and the default values from the system.

The options which apply to the useradd command are:

-c comment

The new user’s password file comment field.

-d home\_dir

The new user will be created using home\_dir as the value for the user’s login directory. The default is to append the login name to default\_ home and use that as the login directory name.

**COMMAND**

login - sign on

**SYNTAX**

login [ name ]

login -p

login -h hostname

login -f name

**DESCRIPTION**

login is used when signing onto a system. It can also be used to switch from one user to another at any time. If an argument is not given, login prompts for the username.

**COMMAND**

passwd - update a user’s authentication token(s).

**SYNTAX**

passwd [-k] [-l] [-u [-f]] [-d] [-n mindays] [-x maxdays] [-w warndays] [-i

inactivedays] [-S] [username]

**OPTIONS**

-k The option, -k, is used to indicate that the update should only be for expired authentication tokens(passwords); the user wishes to keep their non-expired tokens as before.

-l This option is used to lock the specified account and it is

available to root only.

**COMMAND**

hostname - show or set the system\x{2019}s host name

**SYNTAX**

hostname [-v] [-a] [--alias] [-d] [--domain] [-f] [--fqdn] [-i] [--ip-

address] [--long] [-s] [--short] [-y] [--yp] [--nis] [-n] [--node]

hostname [-v] [-F filename] [--file filename] [hostname]

**DESCRIPTION**

Hostname is the program that is used to either set or display the current host, domain or node name of the system. These names are used by many of the networking programs to identify the machine. The domain name is also used by NIS/YP.

**GET NAME**

When called without any arguments, the programs displays he current names:hostname will print the name of the system as returned by the gethostname(2) function.

**COMMAND**

logout - sign out

logout [status] - Exit the shell, returning status to invoking program if

Specified. Can be used only in a login shell. Otherwise, use exit.

**COMMAND**

shell - creates and manipulate a shell widget.

**SYNTAX**

shell pathName ? options?

**INHERITANCE**

itk::Toplevel <- shell

**STANDARD OPTIONS**

background cursor foreground

**DESCRIPTION**

The shell command creates a shell which is a top level which supports modal operation.

**METHODS**

The shell command creates a new Tcl command whose name is pathname. This command may be used to invoke various operations on the widget.

**DIRECTORY AND FILE RELATED COMMANDS**

**COMMAND**

mkdir - make directories

SYNTAX

mkdir [OPTION] DIRECTORY...

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options

too.

-m, --mode=MODE

set permission mode (as in chmod), not rwxrwxrwx - umask

-p, --parents

no error if existing, make parent directories as needed.

**COMMAND**

cd [-L|-P][dir]

Change the current directory to directory. The variable HOME is the default dir. The variable CDPATH defines the search path for the directory containing dir, Alternative directory names in CDPATH are separated by a colon (:)

**COMMAND**

cat - concatenate files and print on the standard output

SYNTAX

cat [OPTION] [FILE]...

DESCRIPTION

Concatenate FILE(s), or standard input, to standard output.

-A, --show-all

equivalent to -vET

-b, --number-nonblank

number nonblank output lines

**COMMAND**

pwd - print name of current/working directory.

SYNTAX

pwd [option]

Print the full file name of the current working directory.

**COMMAND**

file - determine the file type

SYNTAX

file [-bciknsvzL][-f namefile][-m magicfiles] file

file -C [-m magicfile]

DESCRIPTION

File tests each argument in an attempt to classify it. There are three sets of tests, perform in this order: file system tests, magic number tests and language tests. The first test that succeeds causes the file type to be printed.

**OPTIONS**

-b do not prepend file names to output lines(brief mode).

-c cause a checking printout of the parsed form of the magic file. This

is usually used in conjunction with –m to debug a new magic file

before installing it.

-C write a magic.mgc output file that contains a pre parsed version of

file.

**COMMAND**

tree - list contents of directories in a tree-like format.

SYNTAX

tree [-adfghilnpqstuxACDFN][-P pattern][-I pattern][directory]

DESCRIPTION

Tree is a recursive directory listing program that produces a depth intended listing of files. Upon completion of listing all files/directories found, tree returns the total number of files and/or directories listed.

**COMMAND**

cp - copy files and directories

SYNTAX

cp [OPTION]... SOURCE DEST

cp [OPTION]... SOURCE... DIRECTORY

cp [OPTION]... --target-directory=DIRECTORY SOURCE...

DESCRIPTION

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY. Mandatory arguments to long options are mandatory for short options too.

-a, --archive

same as -dpR

--backup[=CONTROL]

make a backup of each existing destination file

-b like --backup but does not accept an argument

**COMMAND**

mv - move (rename) files

SYNTAX

mv [OPTION]... SOURCE DEST

mv [OPTION]... SOURCE... DIRECTORY

mv [OPTION]... --target-directory=DIRECTORY SOURCE...

DESCRIPTION

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

**COMMAND**

more - file perusal filter for crt viewing

SYNTAX

more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]

DESCRIPTION

More is a filter for paging through text one screen full at a time. This

version is especially primitive. Users should realize that less (1) provides more (1) emulation and extensive enhancements.

**OPTIONS**

-num This option specifies an integer which is the screen size (in lines).

-d more will prompt the user with the message "[Press space to continue, ‘q’ to quit.]" and will display "[Press ‘h’ for instructions.]"instead of ringing the bell when an illegal key is pressed.

-l more usually treats ^L (form feed) as a special character, and will pause after any line that contains a form feed. The -l option will prevent this behavior.

**COMMAND**

rm - remove files or directories

SYNTAX

rm [OPTION]... FILE...

DESCRIPTION

This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories.

If a file is unwritable, the standard input is a tty, and the -f or --force option is not given, rm prompts the user for whether to remove the file. If the response does not begin with ‘y’ or ‘Y’, the file is skipped.

**OPTIONS**

Remove (unlink) the FILE(s).

-d, --directory

unlink FILE, even if it is a non-empty directory

(super-user only)

-f, --force

Ignore nonexistent files, never prompt

-i, --interactive

Prompt before any removal.

**COMMAND**

man - format and display the on-line manual pages

manpath - determine user’s search path for man pages

SYNTAX

man [-acdfFhkKtwW] [--path] [-m system] [-p string] [-C config\_file]

[-M pathlist] [-P pager] [-S section\_list] [section] name ...

DESCRIPTION

man formats and displays the on-line manual pages. If you specify section, man only looks in that section of the manual. name is normally the name of the manual page, which is typically the name of a command, function, or file. However, if name contains a slash (/) then man interprets it as a file specification, so that you can do man. /foo.5 or even man /cd/foo/bar.1.gz.

**OPTIONS**

-C config\_file

Specify the configuration file to use; the default is

/etc/man.config.

-M path

Specify the list of directories to search for man pages. Sepa-

rate the directories with colons. An empty list is the same as not specifying -M at all. See SEARCH PATH FOR MANUAL PAGES.

**WILDCARDS**

File names are the most common arguments used in a command. Often, you may know only part of the file name or you may want to references several file names that have the same extension or begin with the same characters. The shell provides a set of special characters called wildcards that search out, match and generate a list of file names. Wild characters are asterisk, the question mark and brackets (\*,?, []) .

WILDCARD SYMBOLS

\* Match on any set of characters in file names

? Match on any single character in file name

[ ] Match on class

**SEARCH PATH FOR MANUAL PAGES**

man uses a sophisticated method of finding manual page files, based on the invocation options and environment variables, the /etc/man.config configuration file, and some built in conventions and heuristics.

First of all, when the name argument to man contains a slash (/), man assumes it is a file specification itself, and there is no searching involved.

PATHNAME:

Specifies the path of the current working directory. Suppose that ‘data’ is the current working directory, then we have specifies it ‘/home/data’.

RELATIVE PATHNAME:

Specify only the full path name from the current directory

Eg- To refer directory’ data’, we just specify only’/data’.

ABSALUTE PATH NAME

Specify the full path name. Suppose, ‘data’ contained in dir. ‘Linux’, then to specify ‘data’, we use the full path as follows,

‘/home/Linux/data’.

**REFERRING TO HOME DIRECTORIES**

1: $ :HOME: Environment variable store our home directory name.

2: ~: The tilde represents our home directory name.

3:’.’ : Single dot represent current working directory.

4:’**..** ’: Double dot refer to a directly above the current directory.

5:OLD PWD: It refers to the previous working directory before you change to the new one.

**FILES AND DIRECTORIES**

root

mnt

dev

etc

bin

sbin

user

tmp

root

home

var

/bin: Contain Common LINUX user commands such as ls, date and chmod.

/dev: Contains files representing access points to device on your systems.

/etc: Contains administrative configuration files.

/home: Contains directories assigned to each user with a log in account.

/mnt: Provides allocation for mounting devices such as remote file system and

removable media.

/root: Represents root user’s home directory.

/sbin: Represents administrative files and the daemon process.

/tmp: Contains temporary files used by the applications.

/usr: Contains user documentation, games, graphical files(x11), libraries

(libo) and a variety of other user and administrative commands and files.

/boot: Have the bootable Linux kernel and boot loader configuration files.

/var: Contains directories of date used by various applications. In particular,

this is where you would place file that you share as an FTP server or a

web server.

**CHANGING PERMISSIONS**

**NAME**

chmod – change file access permissions

SYNTAX

chmod [OPTION]…MODE[,MODE]…FILE…

chmod [OPTION]…OCTAL-MODE FILE…

chmod[OPTION] …--reference=RFILE FILE…

DESCRIPTION

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions.chmod never changes the permissions of symbolic links; the chmod system call can not change their permissions. This is not a problem since the permissions of symbolic links are never used. However, each symbolic link listed on the command line, chmod changes the permissions of the pointed-to file. In contrast, chmod ignores symbolic links encountered during recursive directory traversals.

**OPTION**

Change the mode of each FILE to MODE.

-c, --changes

like verbose but report only when a change is made

-f, --silent, --quiet

like suppress most error messages

-v, --verbose

output a diagnostic for every file processed

-R, --recursive

change files and directories recursively

--help

display this help and exit

--version

output version information and exit

**Vi EDITOR**

Although many different kinds of editors may be available on any given Linux system, all systems have the two standard editors: Ed and Vi .Vi, which stands for “visual”, remains one of the most widely used editors in Linux. The vi editor displays a wall screen of data at a time and allows you to edit any data shown on the screen.

Editors use a key board for two very different operations: to specify editing commands and to receive character input. As editing commands, certain keys will perform deletions, some will execute changes, and others will perform cursor movement. Instead of dividing the command and input functions among different keys, the Vi editor has two separate modes of operations for the key board: command mode and input mode. In command mode, all the keys on the key board become editing commands. In the input mode, the keys on th4e key board become input characters.

When you change modes, the functionality of the key board changes. When you invoke the Vi editor, you are placed in command mode. Each key know becomes an editing command.

Once in the input mode, the key board again changes functionality. Each key now represents a character to be input to the text. The keyboard becomes like a type writer.

Though the Vi command mode handles most editing operations, there are some, such as file saving and global substitutions, that it cannot perform. for such operations you need to execute line editing commands. you enter the line editing mode using the Vi colon command,:. The colon is a special command that allows you to perform one line editing operation. Upon pressing the colon, a line opens up at the bottom of the screen with the cursor placed at the beginning of the line.you are know in the line editing mode. In this mode,you enter an editing command on a line, press ENTER, and the command is executed. Entry into this mode is only temporary. Upon pressing ENTER, you are automatically reurned to the Vi command mode, and the cursor returns to its previous position on this screen.

The command and input operations constitute two very separate modes.Add to this line editing mode,and you are faced to with three very different modes of operations in Vi.The line editing mode operates on the line : you type in a command with its arguments and terminate the command the command by pressing ENTER.The Vi command mode,however ,operates by single keys.Simply pressing a key or sequence of keys executes an editor command.The Vi input mode input characters into a text file.Any key is a valid character for except for ESC.The ESC returns you to the command mode.The Vi editor you can create,save,close and quit files.The commands for each are not all that similar. Saving and Quiting a file involves the use of special line editing commands, where as closing a file is a Vi editing command. Creation of a file is usually specified on the same shell command line that invokes the Vi editor.

To editor file,type Vi and the name of the file on the shell command line.If a file by that name does not exist the system will create it.In effect ,giving the name of the file that does not exist instructs the Vi editor to create that file.

On Linux vi command is usually a link to one of these programs.

Vi filename

Once the file is opened, you are in command mode. After all there are three modes together.

* Command mode
* Insert mode
* Exmode

The vi program has 3 modes of operation:

**Command Mode:** Here all the keys pressed by the user are interpreted to be

editor commands

**Insert Mode:** It permits us to give commands at the command line. The

bottom line of the vi screen is called the command line.

Key Cursor Movement

h Moves cursor left one character

l Moves cursor right one character

k Moves cursor up one line

j Moves cursor down one line

w Moves cursor forward one line

W Moves cursor forward one space delimited word

b Moves cursor backward one word

B Moves cursor back one space delimited word

e Moves cursor to the end of the next word

E Moves cursor to the end of the next space delimited word

o Moves cursor to the beginning of the line

$ Moves cursor to the end of the line

ENTER Moves cursor to beginning of the next line

~ Moves cursor to beginning of previous line

( Moves cursor to beginning of sentence

) Moves cursor to end of sentence; successive command moves to beginning of next sentence

{ Moves cursor to beginning of paragraph

} Moves cursor to end of paragraph

CTRL-F Moves forward by a screen of text; the next screen of text is displayed

CTRL-B Moves backward by a screen of text; the previous screen is displayed

CTRL-D Moves forward by one-half screen of text

CTRL-U Moves backward by one-half screen of text

G Moves cursor to last line in the text

numG Moves cursor to specific line number 45G will place the cursor on line 45

H Moves cursor to line displayed on screen

M Moves cursor to middle line displayed on screen

,, Moves the cursor to its previous location in the text

mmark Places a mark on a line of text; the mark can be any alphabetic character

’mark Moves the cursor to the line with the mark

**Input** All input commands place the user in input; the user leaves input with ESC

a Enters input after the cursor

A Enters input at the end of a line

i Enters input before the cursor

I Enters input at the beginning of a line

o Enters input below the line the cursor is on; insert a new empty line below the one the cursor is currently on

O Enters input above the line the cursor is on; insert a new empty line above the one the cursor is currently on

**Delete**

x Deletes the character the cursor is on

X Deletes the character before the character the cursor is on

dw Deletes the word the cursor is on

db Deletes the beginning of a word

dW Deletes space delimited word

dB Deletes to beginning of a space delimited word

dd Deletes the line the cursor is on

D Deletes the rest of the line the cursor is on

d0 Deletes text from cursor to beginning of line

d Deletes following text specified

d) Deletes rest of a sentence

d} Deletes rest of a paragraph

dG Deletes rest of the file

dm Followed by a mark, deletes everything to mark

dL Deletes the rest of screen

dH Deletes top of the screen

J Joins the line below the cursor to the end of the current line; in effect, deleting the new line character of the line the cursor is on

**Change** Except for the replace command, x, all commands place the user into input after deleting text

sDeletes the character the cursor is on and place the user into the input mode

cw Deletes the word the cursor is on and place the user into the input mode

cb Changes to beginning of a word

cW Changes space delimited word

cB Changes to beginning of a space delimited word

cc Deletes the line the cursor is on and place the user into input

C Deletes the rest of the line the cursor is on and place the user into input

c0 Changes text from cursor to beginning of line

c Changes following text specified

c) Changes the rest of a sentence

c} Changes the rest of a paragraph

cG Changes the rest of file

cm Followed by a mark, changes everything to mark

cL Changes the rest of the screen

cH Changes the top of the screen

r Replace the character the cursor is on; after pressing **x** the user enters the replacement character; the change is made without entering input; the user remains in the Vi command mode

R First places into the input mode, then overwrites character by character; appears as an overwrite mode on the screen but actually is in input mode

**Move** Moves text by first deleting it, moving the cursor to desired place of insertion, and then pressing the **p** command. (When text is deleted, it is automatically held in a special buffer.)

P Inserts deleted or copied text after the character or line the cursor is on

P Inserts deleted or copied text before the character or line the cursor is on

dw p Deletes a word, then moves it to the place you indicate with the cursor (press **p** to insert the word  *after*  the word the cursor is on)

dw p Deletes a word, then moves it to the place you indicate with the cursor (press p to insert the word  *before*  the word the cursor is on

dd p Deletes a line, then moves it to the place you indicate with the cursor (press **p** to insert the word  *after*  the line the cursor is on)

d p Deletes following text specified, then moves it to the place you indicate with the cursor( press **p** or **P**)

d) p Moves the rest of a sentence

d} p Moves the rest of a paragraph

dG p Moves the rest of the file

dm p Followed by a mark, moves everything to mark

dL p Moves the rest of the screen

**File Operation Effect**

W Write Saves file

r filename Read Inserts file text

q Quit Quits editor

**Delete, Move,**

**And Copy**

d Delete Deletes a line or a set of lines

m*Num* Move Move a line or a set of lines by deleting

them and then inserting them after line *Num*

co*Num* Copy Copies a line or a set of lines by copying

them and then inserting the copied text after

line *Num*

**Line Reference Description**

*Num* Line number A number references that line number

*Num, Num* Set of lines Two numbers separated by a Comma references a set

of lines

**Special Characters**

Any character Matches on any one possible character in a pattern

\*Repeated chars Mathes on repeated characters in a pattern

[ ] Classes Matches one classes of characters, a set of characters, in a

pattern

**^** Start of a line References the beginning of a line

$ End of a line References the end of a line

/< Start of a word References the start of a word

>/ End of a word References the end of a word

**Copy** Copy commands are mean to be used in conjunction with the p command. Upon copying text, the user moves the cursor to the place where the copy is to be inserted; the p command then inserts the texts after the character or line the cursor is on

ywCopies the word the cursor is on,then moves the word to the place you indicate with the cursor (press p to insert after the word the cursor is on).

yb Copies to beginning of a word

yW Copies space delimited word

yB Copies to beginning of a space delimited word

yy or Y Cop[ies the line the corsor is on,then moves the line to the place you indicate with the cursor (press p to insert after the line the cursor is on)

y Copies following text specified

y) Copies the rest of a sentence

y} Copies the rest of a paragraph

yG Copies therest of the file

ym Followed by a mark ,copies everything to mark.

yL Copies the rest of the screen

yH Copies to the top of the screen

**THE LINUX SHELLS**

The shell is a program that acts as a buffer between you and the operating system. In its role as a command interpreter, it should act invisibly. It also can be used for simple programming.

**PURPOSE OF A SHELL**

There are 3 main uses for the shell,

1. Interactive use.
2. Customization of your Linux session.
3. Programming.

|  |  |
| --- | --- |
| Program name | Shell |
| /bin/sh  /bin/bash  /bin/csh  /bin/tcsh | Bourne again shell  Bourne again shell  C shell or [tcsh]  tcsh |

**Shell flavors**

Many different Linux shells are available:

* the Bourne-Again shell [bash] that is based on the Bourne shell [sh]

and is standard for Linux.

* the C shell [csh] that uses c syntax and has many conveniences.
* tcsh, an extension of the csh that appears instead of csh in many Linux distributions.

**Bash: The Bourne again shell**

Bash is the GNU version of the standard Bourne shell- the original Unix shell – and incorporates many popular features from the other shells such as sh, tcshand the Korn shell [ksh]. If execute as part of the user’s login, bash starts by executing any commands found in ~/.bash\_profile, or ~/.bash\_login.

**csh and tcsh**

on some versions of Linux, tcsh is used as C shell.

**MOUNTING FILE SYSTEMS**

COMMAND

mount - mount the file system.

SYNTAX

mount [-lhV]

mount -a [-fFnrsvw] [-t vfstype] [-O optlist]

mount [-fnrsvw] [-o options[,….]] device | dir

mount [-fnrsvw] [-t vfstype] [-O options] device dir

DESCRIPTION

All files accessible in a UNIX system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree. Conversely, the ‘umount’ command will detach it again.The standard form of mount command, is

mount -t type device dir

**OPTIONS**

-V output version.

-h print a help message.

-v verbose mode.

-a mount all file systems (of the given types) mentioned in fstab.

-F (used in conjunction with –a). Fork off a new incarnation of mount for each device of possible characters in a file name.

**COMMAND**

umout-Unmount the file system

SYNTAX

Umount [-dflnrv][-t vfstype][-o option]

DESCRIPTION

The umount command detaches the file system(s) mentioned from the file heirachy. A file system is specified by giving the directory where it has been mounted.

**OPTION**

-n Umount without writing in /etc/mtab

-r In case unmounting fails, try to remount read-only.

**FILTER**

Filters are commands that read data, perform operations on that data, and then send result to the standard output. Filters generate different kinds of output, depending on their task. Filter can be operating on a stream of data. As data is passed through the filter, it is analyzed, screened or modified. The data output stream to a filter consists of a sequence of bytes that can be received from the files, devices or a output of other commands or filters. The filter operates on a stream of a data, but it does not operate (or modify) the source of data. Only files data is read and fed into filter. The output of a filter is usually sent to the standard output. It can be then directed to the file or device, or piped as input to another utility or filter.

A filter is any command that gets its input from standard input, manipulates the lines, and then send them to standard output. Three filters – grep, sed, awk, - are so powerful. There are some common filters: cat, cmp, comm., cut, diff, head, paste, sort, tail, tr, unique and wc.

**COMMON FILTERS**

**FilterAction**

more passes all data from input to output, with pauses at the end of each screen of a data

cat passes all data from input to output

cmp compares two files

comm. Identifies common lines in two files

cut passes only specified columns

diff identifies differences between two files or between common files in two

directories

head passes the number of specified lines at the beginning of data

paste combines columns

sort arranges the data in sequence

tail passes the number of specified lines at the end of the data

tr translate one or more characters as specified

uniq deletes duplicate (repeated) lines

wc counts characters, words or lines

sed passes edited lines

awk passes edited lines – passes lines

**INPUT OUTPUT REDIRECTION**

**Standard input /output and redirection**

When UNIX was designed, a decision was made to distinguish between the physical implementation and logical organization of a file. This logical file organization extends to input and output operations. The data in input and output is organized like file. Data input at the keyboard is placed in a data stream arranged as a set of bytes. Data output from a command or program is also placed in a data stream and arranged as a continuous set of bytes. This input data stream is referred to in Linux as the standard input and output data stream is called the standard output. Because the standard input and standards output have the same organization as that of a file. Linux as redirection capability that lets you easily move data in out of files.

**Redirecting the standard output :> and >>**

If you want to direct the standard output to a file rather than the screen to this you place the output redirection operator ,> and name of a file on the command line after the linux command

$ cat myletter > newletter

The redirection operation creates the new destination file. If the file already exists it will be over written with the data in the standard output.

**Appending the standard output :>>**

We can also append the standard output to an existing file using the >> redirection operator.Instead of overwriting the file , the data in the standard output is added at the end of the file.

$ cat myletter >> alletters

$ cat oldletter >> alletters

**Redirecting the standard input:<**

The standard input may be received from the file rather than the keyboard. The operator for redirecting the standard input is less than sign, <.

$ cat < myletter

don Bosco college, angadikadavu

$

**Using redirection and pipes with filters**

Filters send their output to the standard output and so, by default, they display their output on the screen. The simplest filters merely output contents of files. We can save the output of a filter in a file using redirection or pipes. Save for e. g.: to print we use

$ cat complist | Ipr

**PIPING**

The UNIX piping facility let us connect commands to other commands. This facility is of at most importance in combining Unix commands and operations if can be really useful to redirect the output of one program so that it become the input of another program, there by joining two programs to send the output of one command as input of another the two command must be joined using a pipe ( | ) character.The pipe operator,|, placed between two command form a connection between them.

$ ls \*.c | lpr :-To generate the list of file names with .c extension, then this list is piped to the lpr (to the printer) command.

$ cat | lpr

This text will be printed ^D$ :-This example cat first read the input from the keyboard instead of file and pipes the output to the lpr command.

**Pipes and Redirection : tee**

If you want to redirect the standard output to a file and, at the same time, display the content of the output on the screen so that you can see what you are saving. You can do this with tee command.

$ sort mylist | tee sfile

computer

modem

screen

$

**SHELL PROGRAMS**

**1.Find the factorial of a given number**

**Aim:**

Create a program to find the factorial of a given number

**Algorithm:**

Step1 : start

Step2 : read number n

Step3 : set f=1 and j=1

Step4 : if j less than n go to step 5 otherwise go to step 6

Step5 : fact=fact\* j , j=j+1 go to step 4

Step6 : print value of f

Step7 : stop

**Program:**

echo "Enter the number"

read n

f=1

j=1

while [ $j -le $n ]

do

f=`expr $f \\* $j`

j=`expr $j + 1`

done

echo "Factorial = $f"

**Output:**

Enter the number 5

Factorial = 120

.

**2.Find Fibonacci Series**

**Aim:**

Create a program to print Fibonacci series up to a given number entered as command line

**Algorithm:**

Step 1 : start

Step 2 : read n

Step 3 : if n eqyal to 0 go to step 4 otherwise go to step 5

Step 4 : print value 0

Step 5 : set a = 0 and b =1 k =1

Step 6 : print a and b

Step 7 : set i = n-2

Step 8 : if k less than i go to step 9 otherwise go to step 13

Step 9 : set c = a+b

Step 10 : print c

Step 11 : set a = b,b = c

Step 12 : k = k+1 go to step 8

Step 13 : stop

**Program:**

echo "Enter the limit"

read n

if [ $n -eq 0 ]

then

echo 0

exit

fi

a=0

b=1

k=1

echo "The fibnocci series"

echo $a

echo $b

i=`expr $n-2`

while [[ $k -le $i ]]

do

c=`expr $a + $b`

echo $c

a=$b

b=$c

k=`expr $k + 1`

done

**Output:**

Enter the limit

6

The fibnocci series

0

1

1

2

3

5

**3.Find the reverse of a number**.

**Aim:**

Create a program to print the reverse of a number.

**Algorithm:**

Step 1 : start

Step 2 : read number num

Step 3 : set num2=0

Step 4 : if num not equal to 0 go to step 5 other wise go to step 7

Step 5 : calculate num2=num % 10 + num2 \* 10

Step 6 : calculate num=num/10 go to step 4

Step 7 : set print num2

Step 8 : stop

**Program:**

echo “Enter number”

read num

num2=0

echo “Reverse is”

while [ $num -ne 0 ]

do

num2=` expr $num % 10 + $num2 \\* 10 `

num=` expr $num / 10 `

done

echo “$num2”

**Output:**

Enter number

234

Reverse is

432

**4.To check largest among three numbers.**

**Aim :**

Create a program to find the largest among three numbers.

**Algorithm:**

Step 1 : start

Step 2 : read numbers a,b,c

Step 3 : set l=a

Step 4 : if b greater than l go to step 5 otherwise go to step 6

Step 5 : set l=b go to step 8

Step 6 : if c greater than l go to step 7 otherwise go to step 6

Step 7 : set l=c

Step 8 : print value of l

Step 9 : stop

**Program:**

echo “Enter three numbers”

read a b c

l=$a

if [ $b -gt $l ]

then

l=$b

fi

if [ $c -gt $l ]

then

l=$c

fi

echo “Largest among $a $b and $c is $l”

**Output:**

Enter three numbers

23 56 8

Largest among 23 56 and 8 is 56

**5**.**Find even and odd numbers**

**Aim:**

Create a program to find evan and odd numbers.

**Algorithm:**

Step 1 : start

Step 2 : read number n

Step 3 : set b=n%2

Step 4 : if b equal to 0 go to step 5 otherwise go to step 6

Step 5 : print the value of n is even

Step 6 : print the value of n is odd

Step 7 : stop

**Program:**

echo “Enter the number”

read n

b=$[ n%2]

if [ $b -eq 0 ]

then

echo “Given number $n is even”

else

echo “Given number $n is odd”

fi

**Output:**

Enter the number

34

Given number 34 is even

**6. Swapping two numbers**

**Aim:**

Create a program to swap two numbers.

**Algorithm:**

Step 1 : start

Step 2 : read a,b

Step 3 : print the value of a and b

Step 4 : set t=0, t=a, a=b, b=t

Step 5 : print the value of a and b

Step 6 : stop

**Program:**

echo “Enter the values”

read a b

echo “Before swap value of a and b $a $b”

t=0

t=$a

a=$b

b=$t

echo “After swap value of a and b $a $b”

**Output:**

Enter the values

34 45

Before swap value of a and b 34 45

After swap value of a and b 45 34