**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 d from user

Step 5 : print the value of d, right justified in the field of 6 columns

Step 6 : print the value of d, right justified in the field of 2 columns

Step 7 : print the value of d, left justified in the field of 6 columns

Step 8 : print the value of d, 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 3

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

4

Value in a is1

Value in b is 4

Enter a 4 digit number

9876

9876

9876

9876

009876

Enter two floating point number

32.1

56.4

Enter the field width

2

Enter the precision

3

32.100

3.210000e+01

0.00

5.50e-315

Enter a string

chinchu

chinchu

chinchu

chinc

chinchu

chinch

**PROGRAM NO: 2**

AIM : Write a program to reverse a number using typedef

**ALGORITHM**

Step 1: Start

Step 2: Create a user-defined datatype for int using typedef initialize rev=0

Step 3: Read n

Step 4: Check whether n! =0 then goto step 5 otherwise step 9

Step 5: rev=rev\*10

Step 6: rev=rev\*n%10

Step 7: n=n/10

Step 8: goto step 4

Step 9: print rev

Step 10: Stop

**PROGRAM**

#include<stdio.h>

void main()

{

typedef int number;

number n,r,sum=0;

printf("\nPlease enter any number: ");

scanf("%d",&n);

while(n>0)

{

r=n%10;

sum=(sum\*10)+r;

n=n/10;

}

printf("\nReverse is : %d",sum);

}

**OUTPUT**

Please enter any number: 3456

Reverse is :6543

**PROGRAM NO:3**

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

**ALGORITHM**

Step 1: Start

Step 2: Read values of m1, m2, m3, m4 and m5

Step 3: Find average by add m1, m2, m3, m4 and m5 then divide by 5

Step 4: Is average is greater than 90, print A otherwise go to step 5

Step 5: Is average greater than or equal to 80 and less than or equal to 90 print B otherwise go to Step 6

Step 6: Is average is greater than or equal to 70 and less than or equal to 80 print C otherwise go to step 7

Step 7: Is average is greater than or equal to 60 and less than or equal to70 print D otherwise go to step 8

Step 8: Is average is greater than or equal to 50 and less than or equal to60 print E otherwise goto step 9

Step 9: Print „Failed„ and go to step 10

Step 10: 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");

else if(avg>=80&&avg<90)

printf("B+\n");

else if(avg>=70&&avg<80)

printf("C+\n");

else if(avg>=60&&avg<70)

printf("D+\n");

else

printf("fail\n");

}

**OUTPUT**

Enter three marks

51

85

89

C+

**PROGRAM NO :4**

AIM: Program to print numbers equivalent to Roman Numbers using Switch case

**ALGORITHM**

Step 1 : start

Step 2 : declare ch as character

Step 3 : display enter the Roman Character

Step 4 : read ch

Step 5 : select choice ch

Step 6 : when ch equal to I display 1

Step 7 :when ch equal to V display 5

Step 8 :when ch equal to X display 10

Step 9 :when ch equal to L display 50

Step 10 :when ch equal to C display 100

Step 11 :when ch equal to D display 500

Step 12 :when ch equal to F display 100

Step 13 : print “Invalid entry”

Step 14 : stop

**PROGRAM**

#include<stdio.h>

main()

{

char ch;

printf("Enter Roman number \n");

scanf("%c",&ch);

switch(ch)

{

case 'I':

printf("1");

break;

case 'V':

printf("5");

break;

case 'X':

printf("10");

break;

case 'L':

printf("50");

break;

case 'C':

printf("100");

break;

case 'D':

printf("500");

break;

case 'F':

printf("1000");

break;

default:

printf("invalid entry");

}

}

**OUTPUT**

Enter Roman number

X

10

**PROGRAM NO:5**

AIM:Multiplication Table using for loop

**ALGORITHM**

Step 1: Start

Step 2: Read n

Step 3: Set i=1

Step 4: Repeat step 5 and 6 for until i<=10

Step 5: print n, i, n\*i

Step 6: i=i+1

Step 7: Stop

**PROGRAM**

#include<stdio.h>

main()

{

int i,n;

printf("Enter the number:");

scanf("%d",&n);

printf("\nMultiplication table of %d....",n);

for(i=1;i<=10;i++)

{

printf("\n\t%d \* %d = %d",i,n,i\*n);

}

}

**OUTPUT**

Enter the number:

5

Multiplication table of 5....

1 \* 5 = 5

2 \* 5 = 10

3 \* 5 = 15

4 \* 5 = 20

5 \* 5 = 25

6 \* 5 = 30

7 \* 5 = 35

8 \* 5 = 40

9 \* 5 = 45

10 \* 5 = 50

**PROGRAM NO:6**

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 when 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:

800

Armstrong numbers are:

1

153

370

371

407

**PROGRAM NO:7**

AIM:Sum of Natural Number using do while

**ALGORITHM**

Step1: Start

Step 2: Read value of n

Step 3: sum=0

Step 4: i=1

Step 5: Repeat step 6,7 and 8

Step 6: sum=sum+i

Step 7: i=i+1

Step 8: Check whether i<=n then goto step 5

Step 9: print sum

Step 10: Stop

**PROGRAM**

#include<stdio.h>

void main()

{

int i,n,sum=0;

printf("Enter the limit\n");

scanf("%d",&n);

i=0;

do

{

sum=sum+i;

i++;

}

while(i<=n);

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

}

**OUTPUT**

Enter the limit

10

sum=55

**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 :when i greater than n then go to step 5 else go to step 11

Step 5 : read value of a from user

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

Step 7 :when 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 1.73205

**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 : when 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 : when 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>

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: Sum of digits using call by value

**ALGORITHM**

Step 1: start

Step 2: declare n and s

Step 3: read the number n from user

Step 4: call function sumdig(n)

Step 5: print the value sum

Step 6: stop

Function sumgit(num)

Step 1: declare d and sum equal to 0

Step 2: repeat step until num greater than 0

Step 3: set d equal to num % 10

Step 4: set num equal to num / 10

Step 5: set sum equal to sum + d go to step 2

Step 6: return sum

Step 7: stop

**PROGRAM**

#include<stdio.h>

int sumdig(int);

main()

{

int s,n;

printf("Enter a number ");

scanf("%d",&n);

s=sumdig(n);

printf("\nSum of digits of %d=%d",n,s);

}

int sumdig(int num)

{

int d,sum=0;

while(num>0)

{

d=num%10;

num=num/10;

sum+=d;

}

return(sum);

}

**OUTPUT**

Enter a number

369

Sum of digits of 369=18

**PROGRAM NO:11**

AIM: Call 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 : 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: print values a and b

Step 6 : Stop

**PROGRAM**

#include<stdio.h>

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

int a,b;

void main()

{

printf("\nEnter two values\n");

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

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

swap(&a,&b);

}

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

{

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

printf("\n Using call by reference:\nThe swapped values are:\na=%d b=%d",\*a,\*b);

}

**OUTPUT**

Enter two values

6

5

Before swapping

a=6 b=5

Using call by reference:

The swapped values are:

a=5 b=6

**PROGRAM NO : 12**

AIM: Fibonacci series using recursion

**ALGORITHM**

Step 1: start

Step 2: declare n c equal to o and i

Step 3: read number n from the user

Step 4: set i equal to 1

Step 5: when i less than or equal to n then go to step 6 else go to step 8

Step 6: print the value returned by calling function fib(c) and increment the value of c by 1

Step 7: increment the value of i by 1 go to step 5

Step 8: stop

Function fib(n)

Step 1: start

Step 2: when n equal to 0 then return o otherwise goto step 3

Step 3: when n equal to 1 return 1 otherwise go to step 4

Step 4: otherwise return fib(n-1)+fib(n-2)

Step 5: 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: Sum of elements 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: when 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 and sum=0

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

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

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

Step 6: return sum

Step 7: stop

**PROGRAM**

#include<stdio.h>

float fun();

int a[25],i,n;

void main()

{

int 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("sum=%d",s);

}

float fun()

{

int sum=0;

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

{

sum=sum+a[i];

}

return sum;

}

**OUTPUT**

Enter a limit

5

Enter numbers

4

3

2

3

2

sum=14

**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: when 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: when i less than n then go to step 5 else go to step 10

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

Step 6: set large equal to a[i]

Step 7: when 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: when 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: when 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: when i less than n then go to step 5 else go to step 8

Step 5: when 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 in the position %d .\n",i+1);

f=1;

}}

return f;

}

**OUTPUT**

Enter the limit

5

Enter the numbers

4

8

2

1

9

Enter the number to be searched

2

Number found in the position3 .

**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: when 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: when i less than n then go to step 3 else go to step 12

Step 4: set j equal to i+1

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

Step 6: when( 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

10

Enter numbers

34

23

1

2

65

45

29

78

44

8

Sorted numbers are

1

2

8

23

29

34

44

45

65

78

**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: when 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 :when i less than n then go to step 3 else go to step 12

Step 4 : set j equal to 0

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

Step 6 :when ( 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: when 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

10

Enter numbers

23

11

2

56

7

34

56

21

78

90

Sorted numbers are

2

7

11

21

23

34

56

56

78

90

**PROGRAM NO:18**

AIM :Program to find trace of a matrix

**ALGORITHM**

Step 1: start

Step 2:declare a[10][10],i,j,sum=0,m,n as integers

Step 3: display enter the values of m and n

Step 4:read m and n

Step 5:display enter the elements of the matrix

Step 6:for i=0 to m and j=0 to n read the elements of the matrix

Step 7: for i=0 to m and j=0 to ngoto step 8

Step 8: when i = j add array elements to sum

Step 9: display sum

Step 10: stop

**PROGRAM**

#include<stdio.h>

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

int i,j;

void main()

{

int a[10][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");

}

diagonal(a,r,c);

}

void diagonal(int a[][10],intr,int c)

{

int sum=0;

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

{

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

{

if(i==j)

{

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

}

}

}

printf("\nSum of Trace or principle diagonal elements is=%d",sum);

}

**OUTPUT**

Enter row and column of matrix

2

2

Enter the elements of matrix

2

1

2

3

Matrix is

2 1

2 3

Sum of Trace or principle diagonal elements is=5

**PROGRAM NO:19**

AIM: Matrix multiplication

**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 :when r1 not equal to c2 then print multiplication not possible else go tostep 5

Step 5 : set i equal to 0

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

Step 7 : set j equal to 0

Step 8 :when 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: when i less than r1 then go to step 14 else go to step 19

Step 14: set j equal to 0

Step 15: when 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 :when i less than r1 then go to step 4 else go to step 13

Step 4 : set j equal to 0 43

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 :when 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: when i less than m then go to step 15 else go to step 20

Step 15: set j equal to 0

Step 16: when 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);

int i,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(c1!=r2)

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

3

Enter row and column of second matrix

3

2

Enter the elements of first matrix

1

2

3

4

5

6

Enter the elements of second matrix

1

2

3

4

5

6

FIRST MATRIX

1 2 3

4 5 6

SECOND MATRIX

1 2

3 4

5 6

Multiplication table

7 10

19 28

**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 when 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,a;

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

scanf("%s",s);

n=strlen(s);

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

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

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

strcpy(s1,s2);

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

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

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

printf("Comparing the string is\n");

a=strcmp(s3,s4);

if(a==0)

{

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

}

else

{

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

}

strcat(s3,s4);

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

}

**OUTPUT**

Enter the string:

good

Length of string is: 4

Enter two strings:

good

morning

Copied string is:morning

Enter two strings:

good

morning

Comparing the string is

Two strings are not equal

The concatenated string:goodmorning

**PROGRAM NO:21**

AIM: Implementing simple pointer

**ALGORITHM**

Step 1: start

Step 2: declare interger variable i equal to 0 and pointer variable \*p

Step 3: declare floating point variable f equal to 3.4 and pointer variable \*fp

Step 4: declare char variable c equal to 'a' and a pointer \*cp

Step 5: print i,f,c

Step 6: ip equal to &f

Step 7: print value of ip and \*ip

Step 8: fp equal to &f

Step 9: print values of fp and \*fp

Step 10: cp equal to &c

Step 11: print values of cp and \*cp

Step 12: stop

**PROGRAM**

#include<stdio.h>

main()

{

int i=55,\*ip;

float f=45.77,\*fp;

char c='v',\*cp;

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

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

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

ip=&i;

printf("Address of i=%u\n",ip);

printf("Values of i=%d\n",\*ip);

fp=&f;

printf("Address of f=%u\n",fp);

printf("Values of f=%f\n",\*fp);

cp=&c;

printf("Address of c=%u\n",cp);

printf("Values of c=%c\n",\*cp);

}

**OUTPUT**

i=55

f=45.770000

c=v

Address of i=3215420288

Values of i=55

Address of f=3215420284

Values of f=45.770000

Address of c=3215420283

Values of c=v

**PROGRAM NO:22**

AIM: Array of pointers

**ALGORITHM**

Step 1 : start

Step 2 : read the value of n from user

Step 3 : set i equal to 0

Step 4 :when 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 :when i less than n then go to step 9 else go to step 16

Step 9 : set j equal to i+1

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

Step 11: compare &s[i ]and &s[j] when 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]);

}

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:

5

Enter names:

kichu

appu

ammu

devu

anni

Sorted names are:

ammu

anni

appu

devu

kichu

**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

2

2

Sum=4

Difference=0

Product=4

Division=1

**PROGRAM NO: 24**

AIM: Illustrate enumerator datatype

**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 :when sum less than or equal to mathematics then go to step 6 otherwise 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,

physics,

maths,

computerscience

};

main()

{

int marks[5];

enum subjects subject;

printf("Enter marks in five subjects\n");

for(subject=malayalam;subject<=computerscience;subject++)

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

printf("Marks in five different subjects\n");

printf("Malayalam=%d\n", marks[malayalam]);

printf("English=%d\n" ,marks[english]);

printf("Physics=%d\n" ,marks[physics]);

printf("Maths=%d\n" ,marks[maths]);

printf("Computerscience=%d\n", marks[computerscience]);

}

**OUTPUT**

Enter marks in five subjects

45

23

56

78

67

Marks in five different subjects

Malayalam=45

English=23

Physics=56

Maths=78

Computerscience=67

**PROGRAM NO:25**

AIM: Illustrate the concept of bitfield.

**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 temp

{

unsigned int a:3;

unsigned int b:3;

};

main()

{

struct temp t;

int c;

t.a=4;

t.b=6;

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

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

c=t.a+t.b;

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

c=t.a-t.b;

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

c=t.a\*t.b;

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

c=t.a/t.b;

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

}

**OUTPUT**

10

5

Sum=15

Difference=5

Product=50

Division=2

**PROGRAM NO: 26**

AIM:Employee structure using function.

**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 :when s equal to 1 call function read() and go to step 12

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

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

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

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

Step 11 : print “Invalid”

Step 12 :when s less than or equal to 4 go to step 3 otherwise 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 :when 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 :when i less than n then go to step 4 otherwise 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 :when i less than n then go to step 6 otherwise go to step 13

Step 6 :when (e[i].name,m==k) go to step 7 otherwise 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 :when I equal to n goto step 12

Step 12 : print “ not found”

Step 13 : stop

Function sort()

Step 1 : Accessing the members using structure variable

Step 2 :Reapet the steps until i<=n-1;

Step 3 : When it is alphabetically above the goto step 4

Step 4 : Store fist value in temp, second value in first variable and continue the statement

Step 5 : Display the contents

**PROGRAM**

#include<stdio.h>

int n,i,s,c;

char m[10],k[10];

struct employee

{

int empno;

char name[20];

float salary;

}e[10];

void read(struct employee[]);

void display(struct employee[]);

void search(struct employee[]);

void sort(struct employee[],int);

main()

{

do

{

printf("\n 1 ENTER");

printf("\n 2 DISPLAY");

printf("\n 3 SEARCH");

printf("\n 4 SORTED ");

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:

sort(e,n);

break;

default:

printf("invalid\n");

}

printf("\nDo you want to continue? \n1 - CONTINUE\n0 - EXIT\t");

scanf("%d",&c);

}

while(c==1);

}

void read(struct employee e[])

{

printf("Enter limits:\n");

scanf("%d",&n);

printf("Enter Employee Details:\n");

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

{

printf("Enter Employee-Id:");

scanf("%d",&e[i].empno);

printf("Enter Name:");

scanf("%s",e[i].name);

printf("Enter Salary:");

scanf("%f",&e[i].salary);

}

}

void display(struct employee e[])

{

printf("\nEmployee Details:\n");

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

{

printf("Employee-Id:%d\n",e[i].empno);

printf("Name:%s\n",e[i].name);

printf("Salary:%f\n",e[i].salary);

}

}

void sort(struct employee e[],int n)

{

int j,m;

struct employee temp;

printf("\nThe sorted list of names are:");

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

{

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

{

if(strcmp(e[m].name,e[j].name)>0)

{

temp=e[m];

e[m]=e[j];

e[j]=temp;

}

}

}

display(e);

}

void search(struct employee 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("Employee-Id:%d\n",e[i].empno);

printf("Name:%s\n",e[i].name);

printf("Salary:%f\n",e[i].salary);

break;

}

}

if(i==n)

printf("Name not found\n");

else

{

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

}

}

**OUTPUT**

1 ENTER

2 DISPLAY

3 SEARCH

4 SORTED

Enter your choice:

1

Enter limits:

3

Enter Employee Details:

Enter Employee-Id:1

Enter Name:chinchu

Enter Salary:100000

Enter Employee-Id:2

Enter Name:nathiya

Enter Salary:50000

Enter Employee-Id:3

Enter Name:appu

Enter Salary:20000

Do you want to continue?

1 - CONTINUE

0 - EXIT 1

1 ENTER

2 DISPLAY

3 SEARCH

4 SORTED

Enter your choice:

2

Employee Details:

Employee-Id:1

Name:chinchu

Salary:100000.000000

Employee-Id:2

Name:nathiya

Salary:50000.000000

Employee-Id:3

Name:appu

Salary:20000.000000

Do you want to continue?

1 - CONTINUE

0 - EXIT 1

1 ENTER

2 DISPLAY

3 SEARCH

4 SORTED

Enter your choice:

3

Enter the name to be searched:

nathiya

Name found

Employee-Id:2

Name:nathiya

Salary:50000.000000

2

nathiya

50000.000000

Do you want to continue?

1 - CONTINUE

0 - EXIT 1

1 ENTER

2 DISPLAY

3 SEARCH

4 SORTED

Enter your choice:

4

The sorted list of names are:

Employee Details:

Employee-Id:3

Name:appu

Salary:20000.000000

Employee-Id:1

Name:chinchu

Salary:100000.000000

Employee-Id:2

Name:nathiya

Salary:50000.000000

Do you want to continue?

1 - CONTINUE

0 - EXIT 0

ssss

**PROGRAM NO: 27**

AIM:Illustrate the concept of union using function.

**ALGORITHM**

Step 1:Start

Step 2:Read roll no,name,mark

Step 3:Define student record using union

Step 4:Create a student union variable

Step 5:Call student of function

Step 6:stop

Step 7:Student function

Step 8:Start

Step 9:Read the value for each element of the student record

Step 10:Display each element of the student record

Step 11:Stop

**PROGRAM**

#include<stdio.h>

union student

{

int rollno;

char name[20];

int mark;

}s;

void student();

main()

{

student();

}

void student()

{

printf("Enter roll no \n");

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

printf("Rollno is %d \n",s.rollno);

printf("Enter name \n");

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

printf("Name is %s \n",s.name);

printf("Enter mark\n");

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

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

}

**OUTPUT**

Enter roll no

101

Rollno is 101

Enter name

chinchu

Name is chinchu

Enter mark

50

Mark is 50

**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 : when ch equal to 1 then call function insert() and go to step 12

Step 6 : when ch equal to 2 then call function view() and go to step 12

Step 7 : when ch equal to 3 then call function search() and go to step 12

Step 8 : when ch equal to 4 then call function update()and go to step 12

Step 9 : when ch equal to 5 then call function delete() and go to step 12

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

Step 11 : print “exited”

Step 12 : when 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 : when 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 : when 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 : when (fread(&t1,sizeof(t1),1,fp)==1) then go to step 7 elsego 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 : when 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 : when (fread(&t1,sizeof(t1),1,fp)==1) then go to step 8 elsego to step 11

Step 8 : when 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 : when 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 : when 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 : when (fread(&t1,sizeof(t1),1,fp)==1) then go to step 11 else go to step 15

Step 11 : when 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 : when 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 : when 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 : when (fread(&t1,sizeof(t1),1,fp)==1) then go to step 9 elsego to step 11

Step 9 : when 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 : when (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();

void delete();

struct tel

{

int id;

char name[30];

char area[20];

long int 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;

unsigned long l;

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

if(fp==NULL)

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

else

{

fseek(fp,0,2);

l=(unsigned long)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;

unsigned long l;

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

if(fp==NULL)

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

else

{

fseek(fp,0,2);

l=(unsigned long)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;

unsigned long l;

r=sizeof(t);

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

if(fp==NULL)

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

else

{

fseek(fp,0,2);

l=(unsigned long)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);

}

}

}

void delete()

{

FILE \*fp,\*f1;

int n,r,f=0,k;

unsigned long l;

char c;

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

if(fp==NULL)

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

else

{

fseek(fp,0,2);

l=(unsigned long)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:

chinchu

Enter the area

poovanchal

Enter the phone number:9876456876

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

1

Enter the Id:

2

Enter the name:

nathiya

Enter the area:

alakode

Enter the phone number:8301937414

1.INSERT

2.VIE

3.SEARCH

4.UPDATE

5.DELETE

6.EXI

Enter the option:

1

Enter the Id:

3

Enter the name

swathi

Enter the area:

kannur

Enter the phone number:9987654532

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

2

ID NAME AREA Ph Number

1 chinchu poovanchal 9876456876

2 nathiya alakode 8301937414

3 swathi kannur 9987654532

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

3

Enter one ID you want to search:

1

ID NAME AREA Ph Number

1 chinchu poovanchal 9876456876

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

3

Enter one ID you want to search:

3

ID NAME AREA Ph Number

3 swathi kannur 9987654532

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

4

Enter the ID to modified:

1

Enter the new ID:

4

Enter the new name:

NATHIYA.P

Enter the new area:

KOLLAM

Enter the new phone number:

9605215315

Suucefully updated

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

2

ID NAME AREA Ph Number

4

NATHIYA.P KOLLAM 9605215315

2 nathiya alakode 8301937414

3 swathi kannur 9987654532

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

4 NATHIYA.P KOLLAM 9605215315

2 nathiya alakode 8301937414

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

6

**PROGRAM NO: 29**

AIM : Program to append additional item to the 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: when fp equal to null goto step 10 else goto step11

Step10: exit

Step 11: read the position to be updated

Step12: 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

5

Enter empno,name and salary 5 employees

100 chinchu 50000

101 nathiya100000

102 swathi 25000

103 jesna 30000

104 aswathi 40000

Contents of the file emp.txt

100 chinchu 50000.000000

101 nathiya 100000.000000

102 swathi 25000.000000

103 jesna 30000.000000

104 aswathi 40000.000000

Enter recno of the record to be updated

3

Enter new details of the employee

105 aswathi10000

Contents of the file emp.txt

100 chinchu 50000.000000

101 nathiya 100000.000000

105 aswathi 10000.000000

103 jesna 30000.000000

104 aswathi 40000.000000

**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 : when 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 : when 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**

[CHINCHU@DBCPC19 Desktop]$ ./a.out sample.c chinchu nathiya hai

Number of arguments 5

Content of sample.c file

chinchunathiyahai

**Query Set-1**

DML AND DDL COMMANDS

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

chinchudb=# 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

chinchudb=# insert into book values(200,'payyan kadhakal','V.K.N');

INSERT 0 1

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

INSERT 0 1

chinchudb=# insert into book values(400,'indulekha','chandhu menon');

INSERT 0 1

3.Show the contents of the table.

chinchudb=# select \* from book;

id | name | author

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

200 | payyan kadhakal | V.K.N

300 | adu jeevitham | benyamin

400 | indulekha | chandhu menon

(3 rows)

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

chinchudb=# alter table book add price int;

ALTER TABLE

5.Show the contents of the table.

chinchudb=# select \* from book;

id | name | author | price

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

200 | payyan kadhakal | V.K.N |

300 | adu jeevitham | benyamin |

400 | indulekha | chandhu menon |

(3 rows)

6.Update the price of each book

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

UPDATE 1

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

UPDATE 1

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

UPDATE 1

7.Show the content of the table.

chinchudb=# select \* from book;

id | name | author | price

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

200 | payyan kadhakal | V.K.N | 155

300 | adu jeevitham | benyamin | 120

400 | indulekha | chandhu menon | 220

(3 rows)

8.Alter the table by deleting the column author.

chinchudb=# alter table book drop author;

ALTER TABLE

9.Show the content of the table.

chinchudb=# select \* from book;

id | name | price

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

200 | payyan kadhakal | 155

300 | adu jeevitham | 120

400 | indulekha | 220

(3 rows)

10.Delete some rows from book.

chinchudb=# delete from book where id=300;

DELETE 1

11.Show the content of the table.

chinchudb=# select \* from book;

id | name | price

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

200 | payyan kadhakal | 155

400 | indulekha | 220

(2 rows)

12.Drop the book table.

chinchudb=# drop table book;

DROP TABLE

**Query Set-2**

INTEGRITY CONSTRAINTS

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

nathiyadb=# 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

nathiyadb=# insert into depp values(101,'mca',1);

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

3.Show the contents of department table.

nathiyadb=# 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.

nathiyadb=# 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.

nathiyadb=# insert into employ values(201,chinchu','cherupuzha',10000,101);

INSERT 0 1

nathiyadb=# insert into employ values(202,'nathiya','alakode',16000,102);

INSERT 0 1

nathiyadb=# insert into employ values(203,'swathi','anaswara',20000,102);

INSERT 0 1

nathiyadb=# insert into employ values(204,'swetha','mannil',18000,103);

INSERT 0 1

nathiyadb=# insert into employ values(205,'jesna','vellara',18000,101);

INSERT 0 1

6.Show the contents of employee table

nathiyadb=# select \* from employ;

eid | ename | address | salary | dno

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

201 | chinchu | cherupuzha | 10000 | 101

202 | nathiya | alakode | 16000 | 102

203 | swathi | anaswara | 20000 | 102

204 | swetha | mannil | 18000 | 103

205 | jesna | vellara | 18000 | 101

(5 rows)

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

nathiyadb=# 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.

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

ename | dname

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

chinchu | mca

nathiya | biology

swathi | biology

swetha | maths

jesna | mca

(5 rows)

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

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

ename | salary

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

swathi | 20000

(1 row)

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

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

ename

----------

chinchu

nathiya

swathi

swetha

jesna

(5 rows)

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

nathiyadb=# 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

-------

jesna

(1 row)

**Query Set-3**

VIEW

1.create a table project.

chinchudb=# create table project(empno int references empl,p\_name varchar(20),place varchar(20));

CREATE TABLE

2.Insert rows into project table.

chinchudb=# insert into project values(101,'hr','kannur');

INSERT 0 1

chinchudb=# insert into project values(102,'salary','kannur');

INSERT 0 1

chinchudb=# insert into project values(103,'Balance','kannur');

INSERT 0 1

chinchudb=# insert into project values(104,'Project Panchayath','mattannur');

INSERT 0 1 ^

chinchudb=# insert into project values(105,'Project vi','mattannur');

INSERT 0 1

3.Show the content of table project.

chinchudb=# select \* from project;

empno | p\_name | place

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

101 | hr | kannur

102 | salary | kannur

103 | Balance | kannur

104 | Project Panchayath | mattannur

105 | Project vi | mattannur

(5 rows)

4.Create a view in project table.

chinchudb=# create view project\_new as select ename,job,p\_name from empl e ,project p where p.empno=e.empno;

CREATE VIEW

5.Show the contents of view ^

chinchudb=# select \* from project\_new;

ename | job | p\_name

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

chinchu | Accountant | hr

nathiya | Clerk | salary

swathi | Ass.Manager | Balance

jesna | Manager | Project Panchayath

jisna | HR manager | Project vi

(5 rows)

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

chinchudb=# 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.

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

3.Show the contents of project.

chinchudb=# 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

chinchudb=# 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.

chinchudb=# insert into employy values(101,devu','anni',1);

INSERT 0 1

chinchudb=# insert into employy values(103,'kunju',kichu',3);

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

6.Show the contents of employee table.

chinchudb=# select \* from employy;

eid | fname | lname | pid

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

101 | anu | anni | 1

103 | kunju | jacob | 3

102 | ebi | sakharia | 3

104 | neena | roy | 3

105 | zera | elizebath | 5

(5 rows)

7.Create a view.

chinchudb=# 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.

chinchudb=# select \* from projectemployy;

fname | lname | pname | hours

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

anu | anni | hms | 7

kunju | 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=’facascs’

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

fname | lname | pname | hours

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

kunju | jacob | facascs | 13

ebi | sakharia | facascs | 13

neena | roy | facascs | 13

(3 rows)

10.Select value from view .

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

fname | lname | pname | hours

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

anu | anni | hms | 7

(1 row)

**Query Set-4**

AGGREGATE FUNCTIONS

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

chinchudb=# 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.

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

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

INSERT 0 1

3.Show the contents of foodcart.

chinchudb=# 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.

chinchudb=# alter table foodcart add price int;

ALTER TABLE

5. Show the contents of foodcart.

chinchudb=# 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.

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

UPDATE 1

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

UPDATE 1

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

UPDATE 1

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

UPDATE 1

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

UPDATE 1

7.Show the contents of foodcart table.

chinchudb=# 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.

chinchudb=# delete from foodcart where fid=104;

DELETE 1

9. Show the conents of foodcart.

chinchudb=# 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.Retrive number of items in foodcart.

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

count

-------

4

(1 row)

11. Retrive sum of sold in food cart.

chinchudb=# select sum(sold)from foodcart;

sum

-----

237

(1 row)

12.Retrive average price from foodcart.

chinchudb=# select avg(price)from foodcart;

avg

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

20.0000000000000000

(1 row)

13.Retrive maximum price from foodcart.

chinchudb=# select max(price)from foodcart;

max

-----

30

(1 row)

14.Retrive minimum sold from foodcart.

chinchudb=# select min(sold)from foodcart;

min

-----

30

(1 row)

15. Insert values into foodcart.

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

INSERT 0 1

16. Show the contents of foodcart.

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

fname | totalsold

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

sandwitch | 80

burger | 48

pizza | 130

cutlet | 79

(4 rows)

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

fname | avg

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

sandwitch | 25.0000000000000000

pizza | 37.5000000000000000

(2 rows)

**Query Set-5**

SET OPERATIONS

1.Create table depositor.

chinchudb=# create table depositor(id int primary key,name char(20),amount int,branch varchar(25));

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

CREATE TABLE

2. Insert values into depositor.

chinchudb=# insert into depositor values(100,'chinchu',10000,'alakode');

INSERT 0 1

chinchudb=# insert into depositor values(101,'nathiya',12000,'alakode');

INSERT 0 1

chinchudb=# insert into depositor values(102,'swathi',17000,'kannur');

INSERT 0 1

3.Show the contents of depositor.

chinchudb=# select \* from depositor;

id | name | amount | branch

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

100 | chinchu | 10000 | alakode

101 | nathiya | 12000 | alakode

102 | swathi | 17000 | kannur

(3 rows)

4.Create table borrower.

chinchudb=# create table borrower(id int primary key,name varchar(15),amount int,branch varchar(25));

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

CREATE TABLE

5.Insert values into borrower.

chinchudb=# insert into borrower values(110,'appu',100000,'payyannur');

INSERT 0 1

chinchudb=# insert into borrower values(111,'jesna',13000,'malom');

INSERT 0 1

chinchudb=# insert into borrower values(112,'jose',16000,'kannur');

INSERT 0 1

6.Show the contents of borrower.

chinchudb=# select \* from borrower;

id | name | amount | branch

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

110 | appu | 100000 | payyannur

111 | jesna | 13000 | malom

112 | jose | 16000 | kannur

(3 rows)

7. Perform union operation with depositor and borrower table.

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

name

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

jose

jesna

nathiya

swathi

appu

nathiya

(6 rows)

8. Perform intersect operation with depositor and borrower table.

chinchudb=# (select branch from depositor)intersect(select branch from borrower);

branch

----------

kannur

(1 row)

**Query Set-6**

FUNCTION

I.

1. Create table student.

chinchudb=# 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.

chinchudb=# insert into stud values(100,chinchu',67);

INSERT 0 1

chinchudb=# insert into stud values(101,nathiya',85);

INSERT 0 1

chinchudb=# insert into stud values(102,'swathi',55);

INSERT 0 1

chinchudb=# insert into stud values(103,'swetha',65);

INSERT 0 1

chinchudb=# insert into stud values(104,'jesna',78);

INSERT 0 1

3. Show the contents of student table.

chinchudb=# select \* from stud;

id | name | mark

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

100 | chinchu | 67

101 | nathiya | 85

102 | swathi | 55

103 | swetha | 65

104 | jesna | 78

(5 rows)

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

CREATE FUNCTION

5. chinchudb=# 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.chinchudb=# select \* from stud;

id | name | mark

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

100 | chinchu | 67

101 | nathiya | 85

102 | swathi | 55

103 | swetha | 65

104 | jesna | 78

(5 rows)

chinchudb=# select fun1(nathiya');

fun1

------

67

(1 row)

chinchudb=# select fun1(nathiya');

fun1

------

85

(1 row)

chinchudb=# select fun1('swetha');

fun1

------

65

(1 row)

chinchudb=# select fun1('swathi');

fun1

------

55

(1 row)

chinchudb=# select fun1('jesna');

fun1

------

78

(1 row)

II.

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

begin

return a+b;

end

'language'plpgsql';

bash-4.1$ psql chinchudb

chinchudb=# \i funsum.sql

CREATE FUNCTION

chinchudb=# select sum(2,7);

sum

-----

9

(1 row)

chinchudb=# select sum(4,7);

sum

-----

11

(1 row)

**Query Set-7**

TRIGGER

1. Create table student.

chinchudb=# 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.

chinchudb=# insert into stud values(100,'chinchu',67);

INSERT 0 1

chinchudb=# insert into stud values(101,'jisna',85);

INSERT 0 1

chinchudb=# insert into stud values(102,'swetha',55);

INSERT 0 1

chinchudb=# insert into stud values(103,'jesna',65);

INSERT 0 1

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

INSERT 0 1

3. Show the contents of student.

chinchudb=# select \* from stud;

id | name | mark

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

100 | chinchu | 67

101 | jisna | 85

102 | swetha | 55

103 | jesna | 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 chinchudb

CREATE FUNCTION

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

CREATE TRIGGER

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

INSERT 0 1

chinchudb=# select \* from stud;

id | name | mark

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

100 | chinchu | 97

101 | jisna | 115

102 | swetha | 85

103 | jesna | 95

104 | harsha | 108

105 | meera | 78

(6 rows)

**Query Set-8**

MATHEMATICAL FUNCTIONS

MATH FUNCTIONS

1.Sin()

chinchudb=# SELECT sin(90);

sin

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

0.893996663600558

(1 row)

2.cos()

chinchudb=# SELECT cos(0);

cos

-----

1

(1 row)

3.tan()

chinchudb=# SELECT tan(45);

tan

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

1.61977519054386

(1 row)

4.Sqrt()

chinchudb=# SELECT sqrt(25);

sqrt

------

5

(1 row)

5.pow()

chinchudb=# SELECT pow(2,5);

pow

-----

32

(1 row)

6.pi()

chinchudb=# SELECT pi();

pi

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

3.14159265358979

(1 row)

7.radians()

chinchudb=# SELECT radians(90);

radians

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

1.5707963267949

(1 row)

8.log()

chinchudb=# SELECT log(180);

log

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

2.25527250510331

(1 row)

9.abs()

chinchudb=# SELECT abs(-15);

abs

-----

15

(1 row)

10.atan()

chinchudb=# SELECT atan(30);

atan

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

1.53747533091665

(1 row)

11.acos()

chinchudb=# SELECT acos(1);

acos

------

0

(1 row)

12.trunc()

chinchudb=# SELECT trunc(18.23);

trunc

-------

18

(1 row)

13.round()

chinchudb=# SELECT round(45.43);

round

-------

45

(1 row)

14.round()

chinchudb=# SELECT round(45.435310,2);

round

-------

45.44

(1 row)

15.random()

chinchudb=# SELECT random();

random

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

0.13433201238513

(1 row)

16.exp()

chinchudb=# SELECT exp(20);

exp

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

485165195.40979

(1 row)

17.floor()

chinchudb=# SELECT floor(1.01);

floor

-------

1

(1 row)

18.cbrt()

chinchudb=# SELECT cbrt(8);

cbrt

------

2

(1 row)

19.ln()

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

natural\_log

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

3.29583686600433

(1 row)

20.ceil

chinchudb=# SELECT ceil(2.7) ;

ceil

------

3

(1 row)

**Query Set-9**

CHARACTER STRING FUNCTIONS

chinchudb=# SELECT ascii('T');

ascii

-------

84

(1 row)

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

btrim

--------

hello

(1 row)

chinchudb=# SELECT ltrim(' world');

ltrim

-------

world

(1 row)

chinchudb=# SELECT rtrim(' hello ');

rtrim

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

hello

(1 row)

chinchudb=# SELECT initcap('kiran');

initcap

---------

Kiran

(1 row)

chinchudb=# SELECT length('kiran');

length

--------

5

(1 row)

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

substr

---------

loworld

(1 row)

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

substr

--------

world

(1 row)

chinchudb=# SELECT lower('MATHEW');

lower

--------

mathew

(1 row)

chinchudb=# SELECT upper('Joseph');

upper

--------

JOSEPH

(1 row)

chinchudb=# SELECT chr(81);

chr

-----

Q

(1 row)

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

translate

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

Good Evening

(1 row)

**Query Set-10**

TRANSACTION

chinchudb=# 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)

chinchudb=# begin;

BEGIN

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

UPDATE 1

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

UPDATE 1

chinchudb=# delete from library where lid=104;

DELETE 1

chinchudb=# 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)

chinchudb=# commit;

COMMIT

chinchudb=# begin;

BEGIN

chinchudb=# alter table library add copy int;

ALTER TABLE

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

UPDATE 1

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

UPDATE 1

chinchudb=# 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)

chinchudb=# rollback;

ROLLBACK

**Query Set-11**

CURSOR

1. Show the contents of student table.

chinchudb=# select \* from stud;

id | name | mark

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

100 | chinchu | 127

101 | jisna | 145

102 | swathi | 115

103 | swetha | 125

104 | harsh | 138

105 | aleesha| 140

106 | asa | 130

107 | meera | 78

(8 rows)

chinchudb=# begin;

BEGIN

chinchudb=# declare st\_cur cursor for select name from stud;

DECLARE CURSOR

chinchudb=# fetch 5 from st\_cur;

name

--------

chinchu

jisna

swathi

swetha

harsha

(5 rows)

chinchudb=# fetch 3 from st\_cur;

name

---------

Aleesha

asa

meera(3 rows)

chinchudb=# move backward 5 in st\_cur;

MOVE 5

chinchudb=# fetch 3 from st\_cur;

name

---------

swetha

harsha

aleesha

(3 rows)

chinchudb=# fetch next from st\_cur;

name

------

asa

(1 row)

chinchudb=# fetch prior from st\_cur;

name

---------

aleesha

(1 row)

chinchudb=# move forward 5 in st\_cur;

MOVE 2

chinchudb=# fetch 0 from st\_cur;

name

------

(0 rows)

chinchudb=# move forward 5 in st\_cur;

MOVE 0

**Query Set-12**

INDEX

chinchudb=# create index stud\_index on stud(name);

CREATE INDEX

chinchudb=# \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()

**Query Set-13**

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.**

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

chinchudb=# 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

chinchudb=# 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

chinchudb=# 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

chinchudb=# 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

chinchudb=# 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.

chinchudb=# insert into customer values(100,'arnav','bangalore');

INSERT 0 1

chinchudb=# insert into customer values(101,'amitha','kochi');

INSERT 0 1

chinchudb=# insert into customer values(102,'babitha','kochi');

INSERT 0 1

chinchudb=# insert into customer values(103,'praveen','calicut');

INSERT 0 1

chinchudb=# insert into customer values(104,'ambika','trichur');

INSERT 0 1

chinchudb=# select \* from customer;

custno | cname | city

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

100 | arnav | bangalore

101 | amitha | kochi

102 | babitha | kochi

103 | praveen | calicut

104 | ambika | trichur

(5 rows)

chinchudb=# insert into order1 values(200,'03-15-2017',101,35000);

INSERT 0 1

chinchudb=# insert into order1 values(201,'01-12-2016',100,53000);

INSERT 0 1

chinchudb=# insert into order1 values(202,'03-12-2016',104,5600);

INSERT 0 1

chinchudb=# insert into order1 values(203,'12-12-2016',103,45000);

INSERT 0 1

chinchudb=# insert into order1 values(204,'12-09-2016',102,12000);

INSERT 0 1

chinchudb=# 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)

chinchudb=# insert into item values(155,38000);

INSERT 0 1

chinchudb=# insert into item values(156,5600);

INSERT 0 1

chinchudb=# insert into item values(158,12000);

INSERT 0 1

chinchudb=# insert into item values(200,15000);

INSERT 0 1

chinchudb=# insert into item values(202,22000);

INSERT 0 1

chinchudb=# select \* from item;

itemno | unitprice

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

155 | 38000

156 | 5600

158 | 12000

200 | 15000

202 | 22000

(5 rows)

chinchudb=# insert into order\_item values(200,156,1200);

INSERT 0 1

chinchudb=# insert into order\_item values(201,155,100);

INSERT 0 1

chinchudb=# insert into order\_item values(202,158,150);

INSERT 0 1

chinchudb=# insert into order\_item values(203,200,1500);

INSERT 0 1

chinchudb=# insert into order\_item values(204,202,250);

INSERT 0 1

chinchudb=# select \* from order\_item;

orderno | itemno | quantity

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

200 | 156 | 1200

201 | 155 | 100

202 | 158 | 150

203 | 200 | 1500

204 | 202 | 250

(5 rows)

chinchudb=# insert into warehouse values(201,'calicut');

INSERT 0 1

chinchudb=# insert into warehouse values(202,'trivandrum');

INSERT 0 1

chinchudb=# insert into warehouse values(203,'kochi');

INSERT 0 1

chinchudb=# insert into warehouse values(204,'kochi');

INSERT 0 1

chinchudb=# insert into warehouse values(205,'kannur');

INSERT 0 1

chinchudb=# select \* from warehouse;

warehouseno | city

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

201 | calicut

202 | trivandrum

203 | kochi

204 | kochi

205 | kannur

(5 rows)

chinchudb=# insert into shipment values(200,201,'03-23-2017');

INSERT 0 1

chinchudb=# insert into shipment values(201,202,'05-13-2017');

INSERT 0 1

chinchudb=# insert into shipment values(202,203,'05-12-2016');

INSERT 0 1

chinchudb=# insert into shipment values(203,203,'12-12-2016');

INSERT 0 1

chinchudb=# insert into shipment values(203,204,'12-12-2016');

INSERT 0 1

chinchudb=# 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.

chinchudb=# 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)

chinchudb=# delete from item where itemno=158;

DELETE 1

chinchudb=# select \* from item;

itemno | unitprice

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

155 | 38000

156 | 5600

200 | 15000

202 | 22000

(4 rows)

chinchudb=# select s.orderno from shipment s,warehouse w where s.warehouseno=w.warehouseno and w.city='kochi';

orderno

---------

202

203

203

(3 rows)

**Query Set-14**

DEPARTMENT EMPLOYEE RALATION

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.

chinchudb=# 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

chinchudb=# 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.

chinchudb=# insert into dep values(10,'msw','1st floor');

INSERT 0 1

chinchudb=# insert into dep values(11,'bca','2nd floor');

INSERT 0 1

chinchudb=# insert into dep values(12,'ma','1st floor');

INSERT 0 1

chinchudb=# insert into dep values(13,'mcom','3rd floor');

INSERT 0 1

chinchudb=# insert into dep values(14,'bsw','3rd floor');

INSERT 0 1

chinchudb=# insert into empl values(10,'jijo','manager',10,27500);

INSERT 0 1

chinchudb=# insert into empl values(12,'roy','clerk',10,3500);

INSERT 0 1

chinchudb=# insert into empl values(13,'heera','accountant',14,13500);

INSERT 0 1

chinchudb=# insert into empl values(14,'Cristy','treasurer',13,15000);

INSERT 0 1

chinchudb=# insert into empl values(11,'anna','manager',10,37500);

INSERT 0 1

3. Show the the contents of each relation.

chinchudb=# 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)

chinchudb=# 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.

chinchudb=#chinchudb=# 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 orderby salary in ascending order.

chinchudb=# 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.

chinchudb=# select distinct dno from empl;

dno

-----

13

14

10

(3 rows)

7. List the names of employees along with the department name.

chinchudb=# 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)

**Query Set-15**

INSTRUCTOR-DEPARTMENT RELATION

1. Create department realtion.

chinchudb=# 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.

chinchudb=# INSERT INTO dep values('cs', 'chinchu ','kannur');

INSERT 0 1

chinchudb=# INSERT INTO dep values('English','Jisna','chala');

INSERT 0 1

chinchudb=# INSERT INTO dep values('Biothechnology','Swathi','kuthuparamb');

INSERT 0 1

chinchudb=# INSERT INTO dep values('Maths','Aswathi','kuthuparamb');

INSERT 0 1

chinchudb=# INSERT INTO dep values('Malayalam','Jessy','mattannur');

INSERT 0 1

3. Show the contents of department relation.

chinchudb=# SELECT \* from dep;

dep\_name | hod | location

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

cs | chinchu| kannur

English | jisna | chala

Biothechnology | Swathi | kuthuparamb

Maths | Aswathi | kuthuparamb

Malayalam | Jessy | mattannur

(5 rows)

4. Create a instructor realtion.

chinchudb=# 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.

chinchudb=# INSERT INTO instruct values (101,'ammu','cs',20000,'kannur');

INSERT 0 1

chinchudb=# INSERT INTO instruct values (102,'Appu','English',17000,'kannur');

INSERT 0 1

chinchudb=# INSERT INTO instruct values (103,'Aswathi','Biothechnology',30000,'koothuparamb');

INSERT 0 1

chinchudb=# INSERT INTO instruct values (104,'Anjali','Maths',25000,'mattannur');

INSERT 0 1

chinchudb=# INSERT INTO instruct values (105,'Reshma','Malayalam',23000,'mattannur');

INSERT 0 1

6. Show the content of instructor tablr.

chinchudb=# SELECT \* from instruct;

instrct\_id | name | dep\_name | salary | address

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

101 | ammu | cs | 20000.00 | kannur

102 | Appu | 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.

chinchudb=# SELECT name,dep\_name from instruct;

name | dep\_name

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

ammu | cs

Appu | English

Aswathi | Biothechnology

Anjali | Maths

Reshma | Malayalam

(5 rows)

8. Retrieve name and department name from salary where salary greater than 20000.

chinchudb=# 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.

chinchudb=# SELECT avg(salary) from instruct;

avg

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

23000.000000000000

(1 row)

10. chinchudb=# 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. chinchudb=# SELECT name from instruct where salary between 10000 and 20000;

name

--------

ammu

Appu

(2 rows)

12. chinchudb=# DELETE from instruct where dep\_name='cs';

DELETE 1

13. chinchudb=# SELECT \* from instruct;

instrct\_id | name | dep\_name | salary | address

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

102 | Appu | 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. chinchudb=# UPDATE instruct set salary=salary\*1.05;

UPDATE 4

chinchudb=# SELECT \* from instruct;

instrct\_id | name | dep\_name | salary | address

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

102 | Appu | 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.**

chinchudb=# 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

chinchudb=# 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

chinchudb=# 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

chinchudb=# create table owns( driver\_id varchar(10) references car,regno varchar(10)references car);

CREATE TABLE 145

chinchudb=# 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**

chinchudb=#insert into person values('KL13','Chinchu','vallithodu');

chinchudb=#insert into person values('KL14','ammu','peratta');

chinchudb=#insert into person values('KL15','arun','iritty');

chinchudb=#insert into person values('KL16','arjun','perinkary');

chinchudb=#insert into person values('KL17','vivek','peravoor');

chinchudb=# select \* from person;

driver\_id | name | address

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

KL13 | Chinchu | vallithodu

KL14 | ammu | peratta

KL15 | arun | iritty

KL16 | arjun | perinkary

KL17 | vivek | peravoor

(5 rows)

chinchudb=#insert into car values('KR100','BMW',2000);

chinchudb=#insert into car values('KR101','audi',2015);

chinchudb=#insert into car values('KR102','benz',2014);

chinchudb=#insert into car values('KR103','maruthi',2001);

chinchudb=#insert into car values('KR104','BMW',2002);

chinchudb=# select \* from car;

regno | model | year

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

KR100 | BMW | 2000

KR101 | audi | 2015

KR102 | benz | 2014

KR103 | maruthi | 2001

KR104 | BMW | 2002

(5 rows) 146

chinchudb=#insert into accident values(455,'2/4/2015','kannur');

chinchudb=#insert into accident values(456,'4/7/2008','kannur');

chinchudb=#insert into accident values(457,'5/7/2008','kannur');

chinchudb=#insert into accident values(458,'5/9/2009','thalasery');

chinchudb=#insert into accident values(459,'6/9/2009','iritty');

chinchudb=# 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)

chinchudb=#insert into owns values('KL13','KR100');

chinchudb=#insert into owns values('KL14','KR101');

chinchudb=#insert into owns values('KL15','KR102');

chinchudb=#insert into owns values('KL16','KR103');

chinchudb=#insert into owns values('KL17','KR104');

chinchudb=# select \* from owns;

driver\_id | regno

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

KL13 | KR100

KL14 | KR101

KL15 | KR102

KL16 | KR103

KL17 | KR104

(5 rows) 147

chinchudb=#insert into participated values('KL13','KR100',455,50000);

chinchudb=#insert into participated values('KL14','KR101',456,100000);

chinchudb=#insert into participated values('KL15','KR102',457,365420);

chinchudb=#insert into participated values('KL16','KR103',458,30000);

chinchudb=#insert into participated values('KL17','KR104',459,60000);

chinchudb=# 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**

chinchudb=# update participated set damage\_amount=250000 where report\_number=456 and regno='KR101';

UPDATE 1

**4. Show the contents of participated table.**

chinchudb=# 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 148

**5. Find the number of accidents in which cars belonging to a specific model were involved.**

chinchudb=# 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.**

chinchudb=# select count(driver\_id) as people from owns o,car c where c.year=2015 and c.regno=o.regno;

people

--------

1

(1 row) 149

**BANKING ENTERPRISE**

**Consider the following database for a chinchudbing 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.** chinchudb=# 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

chinchudb=# 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

chinchudb=# 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 150

chinchudb=# create table depositor(acc\_no int references account,customer\_name varchar(20) references customer);

CREATE TABLE

chinchudb=# 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

chinchudb=# 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**

chinchudb=# insert into branch values('kannur','kerala',40000);

INSERT 0 1

chinchudb=# insert into branch values('kochi','kerala',90000);

INSERT 0 1

chinchudb=# insert into branch values('bangalore','karnataka',100000);

INSERT 0 1

chinchudb=# insert into branch values('hyderbad','andrapredhesh',80000);

INSERT 0 1

chinchudb=# insert into branch values('alapuzha','kerala',35000);

INSERT 0 1

chinchudb=# insert into account values(101,'kannur',15000);

INSERT 0 1

chinchudb=# insert into account values(102,'kochi',45000);

INSERT 0 1

chinchudb=# insert into account values(103,'bangalore',55000);

INSERT 0 1

chinchudb=# insert into account values(104,'hyderbad',35000);

INSERT 0 1

chinchudb=# insert into account values(105,'alapuzha',22000); 151

INSERT 0 1

chinchudb=# insert into customer values('Chinchu','kannur','kerala');

INSERT 0 1

chinchudb=# insert into customer values('Nathiya','kochi','kerala');

INSERT 0 1

chinchudb=# insert into customer values('Dias','bangalore','karnataka');

INSERT 0 1

chinchudb=# insert into customer values('Dileeep','hyderbad','andrapredhesh');

INSERT 0 1

chinchudb=# insert into customer values('allan','alapuzha','kerala');

INSERT 0 1

chinchudb=# insert into depositor values(101,‟Chinchu‟);

INSERT 0 1

chinchudb=# insert into depositor values(102,‟Nathiya‟);

INSERT 0 1

chinchudb=# insert into depositor values(104,‟Dileeep‟);

INSERT 0 1

chinchudb=# insert into depositor values(105,‟Ebin‟);

INSERT 0 1

chinchudb=# insert into loan values(1001,‟kannur‟,10000.00);

INSERT 0 1

chinchudb=# insert into loan values(1002,‟kochi‟,30000.00);

INSERT 0 1

chinchudb=# insert into loanr values(1003,‟ bangalore‟,50000.00);

INSERT 0 1

chinchudb=# insert into loan values(1004,‟ hyderbad‟,80000.00);

INSERT 0 1

chinchudb=# insert into loan values(1005,‟alapuzha‟,15000.00);

INSERT 0 1 152

chinchudb=# insert into borrower values(‟Chinchu‟,1001);

INSERT 0 1

chinchudb=# insert into borrower values(‟Nathiya‟,1002);

INSERT 0 1

chinchudb=# insert into borrower values(‟Jesna‟,1003);

INSERT 0 1

chinchudb=# insert into borrower values(‟Dileeep‟,1004);

INSERT 0 1

chinchudb=# insert into borrower values(‟Ebin‟,1005);

INSERT 0 1

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

chinchudb=# 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)

chinchudb=# 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) 153

chinchudb=# select \* from depositor;

acc\_no | customer\_name

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

101 | Chinchu

102 | Nathiya

104 | Dileeep

105 | Ebin

(4 rows)

chinchudb=# 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)

chinchudb=# select \* from borrower;

customer\_name | loan\_no

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

Chinchu | 1001

Nathiya | 1002

Dias | 1003

Dileeep | 1004

Ebin | 1005

(5 rows)

chinchudb=# select \* from customer; 154

customer\_name | customer\_street | customer\_city

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

Chinchu | kannur | kerala

Nathiya | kochi | kerala

Dias | bangalore | karnataka

Dileeep | hyderbad | andrapredhesh

Ebin | alapuzha | kerala

(5 rows)

**4. Find all the customers who have an account at *all* the branches located in a specific city..**

chinchudb=# 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

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

Dias

(1 row)

**5. Demonstrate how you delete all account tuples at every branch located in a specific city.**

chinchudb=# 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

chinchudb=# 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) 155

**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.**

chinchudb =# 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

chinchudb =# 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

chinchudb =# 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

chinchudb =# 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 156

chinchudb =# 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.**

chinchudb =# insert into students values('dbc101','Chinchu','major','12/11/1995');

INSERT 0 1

chinchudb =# insert into students values('dbc102','Vinay','major','02/10/1995');

INSERT 0 1

chinchudb =# insert into students values('dbc103','Afna','major','01/03/1995');

INSERT 0 1

chinchudb =# insert into students values('dbc104','Siya','major','12/27/1995');

INSERT 0 1

chinchudb =# insert into students values('dbc105','Edison','major','05/09/1995');

INSERT 0 1

chinchudb =# insert into course values(211,'BCA','CS');

INSERT 0 1

chinchudb =# insert into course values(210,'BCOM','COMMERCE');

INSERT 0 1

chinchudb =# insert into course values(212,'BSc.CS','CS');

INSERT 0 1

chinchudb =# insert into course values(213,'BSc.Maths','MATHS');

INSERT 0 1

chinchudb =# insert into course values(214,'BCA','CS');

INSERT 0 1 157

chinchudb =# insert into enroll values('dbc101',211,2,40);

INSERT 0 1

chinchudb =# insert into enroll values('dbc102',210,1,39);

INSERT 0 1

chinchudb =# insert into enroll values('dbc103',212,2,41);

INSERT 0 1

chinchudb =# insert into enroll values('dbc104',212,2,40);

INSERT 0 1

chinchudb =# insert into enroll values('dbc105',214,2,28);

INSERT 0 1

chinchudb =# insert into text values(124,'Cpp ','DD','Yaswant');

INSERT 0 1

chinchudb =# insert into text values(453,'C','DD','Yaswant');

INSERT 0 1

chinchudb =# insert into text values(765,'QA','DD','Agarwal');

INSERT 0 1

chinchudb =# insert into text values(565,'managment','Fary','Gupta');

INSERT 0 1

chinchudb =# insert into text values(983,'CG','DC','Albert');

INSERT 0 1

chinchudb =# insert into book\_adoption values(211,1,453);

INSERT 0 1

chinchudb =# insert into book\_adoption values(212,2,124);

INSERT 0 1

chinchudb =# insert into book\_adoption values(214,1,565); 158

INSERT 0 1

chinchudb =# insert into book\_adoption values(212,1,765);

INSERT 0 1

chinchudb =# insert into book\_adoption values(210,2,565);

INSERT 0 1

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

chinchudb =# select \* from students;

regno | bame | major | bdate

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

dbc101 | Chinchu | 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)

chinchudb =# 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)

chinchudb =# select \* from text;

book\_isbn | book\_title | publisher | author

-----------+------------+-----------+--------- 159

124 | Cpp | DD | Yaswant

453 | C | DD | Yaswant

765 | QA | DD | Agarwal

565 | managment | Fary | Gupta

983 | CG | DC | Albert

(5 rows)

chinchudb =# 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)

chinchudb =# 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) 160

**4. List any department that has *all* its books published by a specific publisher**

chinchudb =# 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.**

chinchudb =# 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)

**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.

**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. 163

**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 164

**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 165

**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. 166

**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 167

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 168

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 169

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 170

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- 171

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 172

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. 173

/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. 174

**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 175

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. 176

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 177

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 178

**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 179

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

s Deletes 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 180

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 181

**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 182

**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

yw Copies 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 183

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. 186 **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. 186

-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. 187

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 188

**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 189

**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** 190

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

$ 191

**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" 192

**Output:**

Enter the number 5

Factorial = 120

. 193

**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 194

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

5 195

**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” 196

**Output:**

Enter number

234

Reverse is

432 197

**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 198

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 56 199

**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 200

**Output:**

Enter the number

34

Given number 34 is even 201

**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” 202

**Output:**

Enter the values

34 45

Before swap value of a and b 34 45

After swap value of a and b 45 34