C PROGRAMMING

PROGRAM NO: 1

AIM: Simple program using formatted I/O.

ALGORITHM

Step 1 : start

Step 2 : read values of a and b from user

Step 3 : print the values of a and b

Step 4 : read value of n from user

Step 5 : print the value of n, right justified in the field
of 7 columns

Step 6 : print the value of n, right justified in the field
of 3 columns

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

Step 8 : print the value of n, placing zeros before n in the field
of 7 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 strr from user

Step 17 : print the value of strr in the field of 20 columns 2

Step 18 : print the first 10 characters of the string strr in the field
of 30 columns and right justified

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

Step 20 : print the first 10 characters of the string strr in the field of 20 columns and left justified

Step 21 : print the value of strr

Step 22 : stop

PROGRAM

```
void main()
{
int w,p,a,b,c;
float x;
double y;
printf("\nFORMATTED INPUT OUTPUT\n");
printf("\nenter three integer numbers\n");
scanf("%*d%d%d",&a,&b);
printf("\nvalue in a is %d\n value in b is %d\n",a,b);
printf("Enter a 5 digit number\n");
scanf("%5d",&c);
printf("%7d\n",c);
printf("%3d\n",c);
printf("%-7d\n",c);
printf("%07d\n",c);
printf("enter two floating point number\n");
scanf("%f%f",&x,&y);
printf("\n enter the field width\n");
scanf("%d",&w);
printf("\nenter the presition\n");
scanf("%d",&p);
printf("\n%*.*f\n",w,p,x);
printf("%e\n",x);
printf("%-7.2f\n",y);
printf("%10.2e\n",y);
char strr[20];
```

```
printf("\nenter a string\n");
scanf("%s",strr);
printf("%20s\n",strr);
printf("%30.10s\n",strr);
printf("%.5s\n",strr);
printf("%-20s\n",strr);
printf("%5s\n",strr);

}
```

OUTPUT:

```
FORMATTED INPUT OUTPUT

enter three integer numbers
2
3
4

value in a is 3
 value in b is 4
Enter a 5 digit number
23456
  23456
23456
23456
0023456
enter two floating point number
22.3
10.5

 enter the field width
4
```

enter the presition

3

22.300

2.230000e+01

0.00

4.85e-270

enter a string

javaprogram

javaprogram

javaprogra

javap

javaprogram

javaprogram

PROGRAM NO: 2

AIM: Reverse of a number using typedef.

ALGORITHM

step 1: start
step 2: give a new name to int data type using typedef statement
step 3: declare n,rem and set reverse equal to 0
step 4: read the value for reversing
step 5: repeat while n greater than 0
step 6: set rem equal to n % 10
step 7: set reverse equal to reverse * 10 + rem
step 8: set n equal to n /10 goto step 5
step 9: print the vaue reverse
step 7: print the reverse value
step 10: stop

PROGRAM

```
#include<stdio.h>
void main()
{
typedef int number;
number n,reverse = 0,rem;

    printf("Enter the number\n");

    scanf("%d", &n);

    while (n> 0)

    {
```

```
        rem = n % 10;

        reverse = reverse * 10 + rem;

        n /= 10;

    }
    printf(" reverse is  = %d\n",reverse);
}
```

OUTPUT:

```
Enter the number
54321
 reverse is  = 12345
```

PROGRAM NO: 3

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

ALGORITHM

Step 1 : start

Step 2 : read values of mathematics,science,english,physics,chemistry from user

Step 3 : Find percentage by add all the marks then divide by 5

Step 4: if percentege is greater than or equal to 80 then print A+ otherwise go to step 5

Step 5 : if percentage is greater than or equal to 75 then print A otherwise go to step 6

Step 6 : if percentage is greater than or equal to 60 then print B otherwise go to step 7

Step 7 : if percentage is greater than or equal to 45 then print C otherwise go to step 8

step 8 : if percentage is greater than or equal to 35 then print D otherwise go to step 9

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

Step 10 : stop

PROGRAM

```
#include<stdio.h>
void main()
{
 int mathematics,science,english,total,prcnt,physics,chemistry;
 printf("\n Enter marks of 5 subjects each out of 100 ");
 printf("\n\n Maths = ");
 scanf("%d",&mathematics);
 printf("\n Science = ");
 scanf("%d",&science);
 printf("\n English = ");
 scanf("%d",&english);
printf("\n physics =");
scanf("%d",&physics);
```

```
printf("\nchemistry =");
scanf("%d",&chemistry);
 total=mathematics+science+english+physics+chemistry;
 printf("\n Total marks = %d/500",total);
 prcnt=total/5;
 printf("\n\n Percentage = %d",prcnt);
 if(prcnt>=80)
 printf("\n\n Your Grade : A+");
 else if(prcnt>=75)
 printf("\n\n Your Grade : A");
 else if(prcnt>=60)
 printf("\n\n Your Grade : B");
 else if(prcnt>=45)
 printf("\n\n Your Grade : C");
 else if(prcnt>=35)
 printf("\n\n Your grade : D");
 else
 printf("\n\n You Are Failed");
}
```

OUTPUT:

Enter marks of 5 subjects each out of 100

Maths = 30

Science = 40

English = 55

physics =76

chemistry =56

Total marks = 257/500

Percentage = 51

Your Grade : C

Enter marks of 5 subjects each out of 100

Maths = 10

Science = 10

English = 14

physics =13

chemistry =25

Total marks = 72/500

Percentage = 14

You Are Failed

PROGRAM NO: 4

AIM: Find roman letters corresponding number using switch

ALGORITHM

step 1: start
step 2: declar charecter variable r
step 3: read the roman (I,V,X,C,D,F) letter r from user
step 4: if r equal to I then goto step 5 otherwise go to step 6
step 5: print value 1
step 6: if r equal to V then goto step 7 otherwise go to step 8
step 7: print value 5
step 8: if r equal to X then go to step 9 otherwise go to step 10
step 9: print value 10
step 10: if r equal to L go to step 11 otherwise go to step 12
step 11: print value 50
step 12: if r equal to C go to step 13 otherwise go to step 14
step 13: print value 100
step 14: if r equal to D go to step 15 otherwise go to step 16
step 15: print value 500
step 16: if r equal to F go to step 17 otherwise goto step 18
step 17: print value 1000
step 18: print no such roman found
step 19: stop

PROGRAM

```
#include<stdio.h>
void main()
{
char r;
printf("\nenter roman letter(I,V,L,C,D,F)\n");
scanf("%c",&r);
```

```
switch(r)
{
case 'I':
printf("1\n");
break;
case 'V':
printf("5\n");
break;
case 'X':
printf("10\n");
break;
case 'L':
printf("50\n");
break;
case 'C':
printf("100\n");
break;
case 'D':
printf("500\n");
break;
case 'F':
printf("1000\n");
break;
default:
printf("\nno such roman number is found\n");
}
}
```

OUTPUT:

enter roman letter(I,V,L,C,D,F)

D

500

enter roman letter(I,V,L,C,D,F)

g

no such roman number is found

PROGRAM NO: 5

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

ALGORITHM

step 1: start

step 2: read lower and higher limit from user

step 3: repeat while lower less than higher

step 4: set c=lower

step 5: repeat while c>0

step 6: calculate d=c%10

step 7: calculate s=s+(d*d*d) and c=c/10 and to go step 5

step 8: check lower==s go to step 9

step 9: print value of lower

step 10: increment lower by 1 and set s=0 and go to step 3

step 11: stop

PROGRAM

```
#include<stdio.h>
#include<math.h>
void main()
{
 int lower,higher,d,c,s=0;
 printf("\nEnter the lower limit and higher limit:\n");
 scanf("%d%d",&lower,&higher);
 while(lower<higher)
 {
  c=lower;
 while(c>0)
  {
   d=c%10;
   s=s+(d*d*d);
```

```
  c=c/10;
 }
 if(lower==s)
 {
  printf("%d\n",lower);
 }
lower++; s=0;
}
}
```

OUTPUT:

Enter the lower limit and higher limit:

0

1000

0

1

153

370

371

407

PROGRAM NO: 6

AIM: Sum of n natural numbers using do while loop.

ALGORITHM

step 1: start

step 2: read the limit of natural number from the user

step 3: set i=0 and sum=0

step 4: print value of i

step 5: sum=sum+i

step 6: increament i by 1

step 7: repeat step 4 until i<=limit

step 8: print value of sum

step 9: stop

PROGRAM

```
#include<stdio.h>
main()
{
int i,n,sum=0;
printf("Enter the limit  of natural number\n");
scanf("%d",&n);
i=0;
printf("\nnumbers are\n");
do
{
printf("%d\n",i);
sum=sum+i;
i++;
}
while(i<=n);
printf("\n sum=%d",sum);
}
```

OUTPUT

Enter the limit of natural number

5

numbers are

0

1

2

3

4

5

sum=15

PROGRAM NO: 7

AIM: Multiplication table using for lop

ALGORITHM

step 1: start

step 2: declare i and n

step 3: read the number n from user

step 4: set i equal to 1

step 5: if i less than or equal to 10 goto step 6 else goto step 8

step 6: print i,n,i*n

step 7: increment i by 1 goto step 5

step 8: 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:10

Multiplication table of 10....

1 * 10 = 10

2 * 10 = 20

3 * 10 = 30

4 * 10 = 40

5 * 10 = 50

6 * 10 = 60

7 * 10 = 70

8 * 10 = 80

9 * 10 = 90

10 * 10 = 100

PROGRAM NO: 8

AIM: Implementing the break and continue statement.

ALGORITHM

Step 1 : start

Step 2 : read value of n from user

Step 3 : set i equal to 0

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

Step 5 : read value of a from user

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

Step 7 : if a greater than 0 go to step 10 otherwise 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
9
Square root of 9 is 3.000000
```

PROGRAM NO: 9

AIM: Average of n numbers using goto statement.

ALGORITHM

Step 1 : start

Step 2 : read the value of n from user

Step 3 : set i equal to 0

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

Step 5 : read the value of p from user

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

Step 7 : calculate s equal to s+p

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

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

Step 10 : print the value of avg

Step 11 : stop

PROGRAM

```
#include<stdio.h>
void main()
{
    int n,p,i,s=0;
    float avg;
    printf("enter the 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 the limit
10
enter the elements
2
1
2
3
34
3
2
12
2
3
average=6.000000
```

PROGRAM NO: 10

AIM: Program to implement 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("\n sum 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
234

 sum of digits of 234=9
```

PROGRAM NO: 11

AIM: Program to implement 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: Fibanocci series using recursive function.

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: if i less than or wqual to n then go to step 6 otherwise 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: if n equal to 0 then return o otherwise goto step 3

step 3: else if n equal to 1 return 1 otherwise go to step 4

step 4: else return fib(n-1)+fib(n-2)

step 5: stop

PROGRAM

```
#include<stdio.h>
int fib(int);
int main()
{
int n,c=0,i;
printf("enter a positive number");
scanf("%d",&n);
printf("\n the fibonacci series is\n");
for(i=1;i<=n;i++)
{
```

```
printf("%d\n",fib(c));
c++;
}
}

int fib(int n)
{
if(n==0)
return (0);
else if(n==1)
return (1);
else
return(fib(n-1) + fib(n-2));
}
```

OUTPUT:

```
enter a positive number
10

 the fibonacci series is
0
1
1
2
3
5
8
13
21
34
```

PROGRAM NO: 13

AIM: Find sum of elements in an array.

ALGORITHM

Step 1: start

step 2: declare a float function fun(int a[],int n)

step 3: declare integer array val[] variable i and n

Step 4: read the value of n from user

Step 5: set i equal to 0

Step 6: if i less than n then go to step 5 otherwise go to step 7

Step 7: read the value of val[i] from user

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

Step 9: set s equal to call function fun(val,n)

Step 10: print the value of s

Step 11: stop

Function fun(int a[],int n)

Step 1: start

Step 2: set i equal to 0 and sum=0

Step 3: if i less than n then go to step 4 otherwise 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[],int n);
int val[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",&val[i]);
}
s=fun(val,n);
printf("sum=%d",s);
}
float fun(int a[],int n)
{
int sum=0;
for(i=0;i<n;i++)
{
sum=sum+a[i];
}
return sum;
}
```

OUTPUT:

```
sum of array elements
Enter a limit
5
Enter numbers
2
3
6
22
12
sum=45
```

PROGRAM NO: 14

AIM: Find maximum and minimum element in an array.

ALGORITHM

Step 1: start

step 2: declare a function fun(int a[],int n)

step 3: declare integer aray val[] variables n and i

Step 4: read the value of n from user

Step 5: set i equal to 0

Step 6: if i less than n then go to step 5 otherwise go to step 7

Step 7: read the value of val[i] from user

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

Step 9: call function fun(val,n)

Step 10: stop

Function fun(int a[],int n)

Step 1: start

Step 2: set large equal to 0

Step 3: set i equal to 0

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

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

Step 6: set large equal to a[i]

Step 7: if a[i] less than small go to step 8 otherwise 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[],int n);
int val[10],n,i;
void main()
{
printf("Enter limit:\n");
scanf("%d",&n);
printf("Enter numbers:\n");
for(i=0;i<n;i++)
{
scanf("%d",&val[i]);
}
fun(val,n);
}
void fun(int a[],int n)
{
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:

5

Enter numbers:

23

1

22

45

4

Largest number is=45

Smallest number is=1

PROGRAM NO: 15

AIM: Linear search using function.

ALGORITHM

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

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

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

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

Step 7: call function search (b,n)

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

Step 9: Stop

Function search(int a[],int n)

Step 1: start

Step 2: read the value of s from user

Step 3: set i equal to 0

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

Step 5: if a[i] equal to s then go to step 6 otherwise 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 a[],int n);
int f=0,n,b[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",&b[i]);
search(b,n);
if(f==0)
printf("not found\n");
}
int search(int a[],int n)
{
int s;
printf("enter the number to be searched\n");
scanf("%d",&s);
for(i=0;i<n;i++)
{
if(a[i]==s)
{
printf("number found at %d position\n",i+1);
f=1;
}
}
return f;
}
```

OUTPUT:

```
Enter the limit
3
enter the numbers
2
3
1
enter the number to be searched
```

```
1
number found at 3 position
Enter the limit
2
enter the numbers
1
2
enter the number to be searched
3
Not found
Enter the limit
4
enter the numbers
1
2
1
1
enter the number to be searched
1
number found at 1 position
number found at 3 position
number found at 4 position
```

PROGRAM NO: 16

AIM: Implement selection sort using function.

ALGORITHM

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

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

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

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

Step 7: call function selection (a,n)

Step 8: stop

Function selection(a[ ],n)

Step 1: start

Step 2: set i equal to 0

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

Step 4: set j equal to i+1

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

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

Step 7: set temp equal to a[i]

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

Step 9: set a[j] equal to temp

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

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

Step 12: set i equal to 0

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

Step 14: print the value of a[i]

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

Step 16: stop

PROGRAM

```
#include<stdio.h>
void selection(int [],int);
int i,j;
void main()
{
int a[10],n;
printf("Enter limit\n");
scanf("%d",&n);
printf("Enter numbers\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
selection(a,n);
}
void selection(int a[],int n)
{
int temp;
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
printf("sorted numbers are\n");
```

```
for(i=0;i<n;i++)
{
printf(" %d\n",a[i]);
}
}
```

OUTPUT:

```
Enter limit
8
Enter numbers
33
22
1
23
54
26
77
45
sorted numbers are
 1
 22
 23
 26
 33
 45
 54
 77
```

PROGRAM NO: 17

AIM: Bubble sort using function.

ALGORITHM

Step 1: start

Step 2: read the value of n from user

Step 3: set i equal to 0

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

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

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

Step 7: call function bubble(a,n)

Step 8: stop

Function bubble(a[ ],n)

Step 1 : start

Step 2 : set i equal to 0

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

Step 4 : set j equal to 0

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

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

Step 7 : set temp equal to a[j]

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

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

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

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

Step 12: set i equal to 0

Step 13: if i less than n then go to step 14 otherwise 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
5
Enter numbers
21
11
4
66
43
sorted numbers are
 4
 11
 21
 43
 66
```

PROGRAM NO: 18

AIM: Implement 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 : if r1 not equal to c2 then print multiplication not possible else go to step 5

Step 5 : set i equal to 0

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

Step 7 : set j equal to 0

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

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

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

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

Step 12: set i equal to 0

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

Step 14: set j equal to 0

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

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

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

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

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

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

Step 21: stop

Function mul()

Step 1 : start

Step 2 : set i equal to 0

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

Step 4 : set j equal to 0

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

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

Step 7 : set k equal to 0

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

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

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

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

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

Step 13: set i equal to 0

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

Step 15: set j equal to 0

Step 16: if j less than n then go to step 17 otherwise 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(r1!=c2)
printf("not possible\n");
else
{
printf("enter the elements of first matrix\n");
for(i=0;i<r1;i++)
```

```
{
for(j=0;j<c1;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the elements of second matrix\n");
for(i=0;i<r2;i++)
{
for(j=0;j<c2;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("FIRST MATRIX\n");
for(i=0;i<r1;i++)
{
for(j=0;j<c1;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
printf("SECOND MATRIX\n");
for(i=0;i<r2;i++)
{
for(j=0;j<c2;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
mul(a,b,r1,c2);
}
```

```
}
void mul(int a[][10],int b[][10],int r1,int c2)
{
int m[10][10];
printf("multiplication table\n");
for(i=0;i<r1;i++)
{
for(j=0;j<c2;j++)
{
m[i][j]=0;
for(k=0;k<r1;k++)
{
m[i][j]=m[i][j]+(a[i][k]*b[k][j]);
}
printf("%d\t",m[i][j]);
}
printf("\n");
}
}
```

OUTPUT:

```
Enter row and column of first matrix
3
3
Enter row and column of second matrix
3
3
enter the elements of first matrix
2
3
4
2
```

```
3
4
5
4
6
enter the elements of second matrix
2
6
7
6
5
5
4
6
9
FIRST MATRIX
2 3     4
2 3     4
5 4     6
SECOND MATRIX
2 6     7
6 5     5
4 6     9
multiplication table
38      51      65
38      51      65
58      86      109
```

PROGRAM NO: 19

AIM: Program to find trace of a matrix.

ALGORITHM

Step 1 : start

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

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

Step 4 : set i equal to 0

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

Step 6 : set j equal to 0

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

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

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

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

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

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

Step 13: stop

Function diagonal()

Step 1 : start

Step 2 : set sum1 and sum2 equal to 0

Step 3 : set i equal to 0

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

Step 5 : set j equal to 0

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

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

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

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

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

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

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

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

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

Step 15:stop

PROGRAM:

```
#include<stdio.h>
void dioganal(int [][10],int,int);
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");
}
dioganal(a,r,c);
```

```
}
void dioganal(int a[][10],int r,int c)
{
int sum1=0,sum2=0;
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
if(i==j)
{
sum1=sum1+a[i][j];
}
if((i+j)==(r-1))
{
sum2=sum2+a[i][j];
}
}
}
printf("\nSum of Trace or principl dioganal elements is=%d",sum1);
printf("\nSum of off dioganal elements= %d\n",sum2);
}
```

OUTPUT:

Enter row and column of matrix

2

2

enter the elements of matrix

3

2

6

5

matrix is

3 2

6 5

Sum of Trace or principl dioganal elements is=8

Sum of off dioganal elements= 8

PROGRAM NO: 20

AIM: Implementing string operations.

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 s4 from user

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

Step 10 : print "strings are equal"

Step 11 : print "strings are not equal

Step 12 : concatenate strings s3 and s4

Step 13 : print value of s3

Step 14: stop

PROGRAM

```
#include<stdio.h>
#include<string.h>
void main()
{
char s[20],s1[10],s2[10],s3[10],s4[10];
int n,h;
printf("Enter the string:\n");
scanf("%s",s);
n=strlen(s);
printf("****LENGTH****\n");
```

```
printf("Length of string using strlen: %d\n",n);
printf("Enter two strings:\n");
scanf("%s%s",s1,s2);
strcpy(s1,s2);
printf("******COPY*****\n");
printf("Copied string is:%s\n",s1);
printf("Enter two strings:\n");
scanf("%s%s",s3,s4);
printf("*****COMPARE****\n");
h=strcmp(s3,s4);
if(h==0)
{
printf("Two strings are equal\n");
}
else
{
printf("Two strings are not equal\n");
}
strcat(s3,s4);
printf("*****CONCATENATION****\n");
printf("The concatenated string:%s\n",s3);
}
```

OUTPUT:

```
Enter the string:
algorithm
****LENGTH****
Length of string using strlen: 9
Enter two strings:
algorithm
progrmng
******COPY*****
Copied string is:progrmng
Enter two strings:
java
linux
*****COMPARE****
Two strings are not equal
*****CONCATENATION****
The concatenated string:javalinux
```

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("values 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
values 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 unsorted names

Step 3 : set i equal to 0

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

Step 5 : print the value of name[i]

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

Step 7 : set i equal to 0

Step 8 : if i less than 5 then go to step 9 otherwise go to step 16

Step 9 : set j equal to i+1

Step 10: if j less than 5 then go to step 11 otherwise go to step 15

Step 11: compare name[i]and name[j] if it is greater than 0 then go to step 12 otherwise go

to step15

Step 12: set temp equal to name[i]

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

Step 14: set name[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 name[i]

Step 18: stop

PROGRAM:

```
#include<stdio.h>
#include<string.h>
main()
{
char *t,*name[5]={"wicket","xaviour","sterberg","anuradha","baburaj"};
int i,j;
```

```
printf("the unsorted list of five names\n");
for(i=0;i<5;i++)
printf("%s\n",name[i]);
for(i=0;i<5;i++)
for(j=i+1;j<5;j++)
if(strcmp(name[i],name[j])>0)
{
t=name[i];
name[i]=name[j];
name[j]=t;
}
printf("sorted list of names\n");
for(i=0;i<5;i++)
printf("%s\n",name[i]);
}
```

OUTPUT:

```
the unsorted list of five names
wicket
xaviour
sterberg
anuradha
baburaj
sorted list of names
anuradha
baburaj
sterberg
wicket
xaviour
```

PROGRAM NO: 23

AIM: Program to implement 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
5
5
sum=10
Difference=0
product=25
division=1
```

PROGRAM NO: 24

AIM: Program to implement enumerator.

ALGORITHM

Step 1: start

Step 2: declare enumerated data type subject with 5 subjects

Step 3: declare a variable of enumerated data type subject

Step 4: read 5 marks for each subject of enumerated data type subject

Step 5: display the marks of each subject

Step 6: stop

PROGRAM:

```
#include<stdio.h>

enum subjects

{

malayalam,english,physics,maths,computer

};

main()

{

int marks[5];

enum subjects subject;

printf("\nenter the marks of 5 subjects");

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

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

printf("\nmarks of 5 different subject are:");

printf("\nmalayalam:%d",marks[malayalam]);
```

```
printf("\nenglish  :%d",marks[english]);

printf("\nphysics  :%d",marks[physics]);

printf("\nmaths    :%d",marks[maths]);

printf("\ncomputer :%d",marks[computer]);

}
```

OUTPUT:

```
enter the marks of 5 subjects

40

50

43

22

33


marks of 5 different subject are:

malayalam:40

english  :50

physics  :43

maths    :22

computer :33
```

PROGRAM NO: 25

AIM: Program to implement bitfield

ALGORITHM

Step 1 : start

Step 2 : create a structure temp

Step 3 : create structure variable t

Step 4 : set t.a equal to 6

Step 5 : set t.b equal to 3

Step 6 : print the value of t.a

Step 7 : print the value of t.b

Step 8 : calculate operation equal to t.a + t.b

Step 9 : print the value of operation

Step 10 : calculate operation equal to t.a - t.b

Step 11: print the value of operation

Step 12 : calculate operation equal to t.a * t.b

Step 13 : print the value of operation

Step 14 : calculate operation equal to t.a / t.b

Step 15 : print the value of operation

Step 16 : stop

PROGRAM:

```
#include<stdio.h>
struct temp
{
unsigned int a:3;
unsigned int b:3;
};
main()
{
struct temp t;
int operation;
t.a=6;
t.b=3;
printf("%d\t",t.a);
printf("%d\t",t.b);
operation=t.a+t.b;
printf("\n sum           :%d\t",operation);
operation=t.a-t.b;
printf("\n diference      :%d\t",operation);
operation=t.a*t.b;
printf("\n multiplication :%d\t",operation);
operation=t.a/t.b;
printf("\n division       :%d\t",operation);
```

}

OUTPUT:

6 3

sum :9

diference :3

multiplication :18

division :2

PROGRAM NO: 26

AIM: Program to implement structure(employee details).

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 s from user

Step 6 : when s equal to 1 call function read() and go to step 11

Step 7 : when s equal to 2 call function display() and go to step 11

Step 8 : when s equal to 3 call function search() and go to step 11

Step 9 : when s equal to 4 call function sort() and go to step 11

Step 10 : print Invalid

Step 11 : when s less than or equal to 4 go to step 4 otherwise go to step 12

Step 12 : 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 otherwise 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()

Step1:start

Step 2:read j,m

Step 3:create employee structure variable temp

Step 4:for m=0 to n-1 increment m by 1

Step 5:for j=m+1 to n increment j by 1

Step 6:when comparing e[m].name and e[j].name >0 then goto step 7

Step 7:set temp=e[m]

Step 8:set e[m]=e[j]

Step 9:set e[j]=temp

Step 10:call function display

Step11:stop

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 SORT");
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("\n Do you want to continue (1/0)\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 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");
}
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);
```

```
}
```

OUTPUT:

```
1 ENTER
 2 DISPLAY
 3 SEARCH
 4 SORT
enter your choice:
1
Enter limits:
2
Enter Employee Details:
Enter Employee-Id:1
Enter Name:karthik
Enter Salary:29000
Enter Employee-Id:2
Enter Name:albin
Enter Salary:25000

 Do you want to continue (1/0) 1

 1 ENTER
 2 DISPLAY
 3 SEARCH
 4 SORT
enter your choice:
2

Employee Details:
Employee-Id:1
Name:karthik
Salary:29000.000000
Employee-Id:2
Name:albin
Salary:25000.000000

 Do you want to continue (1/0) 1

 1 ENTER
 2 DISPLAY
 3 SEARCH
 4 SORT
enter your choice:
3
Enter the name to be searched:
```

albin
Name found
Employee-Id:2
Name:albin
Salary:25000.000000

Do you want to continue (1/0) 1

1 ENTER
2 DISPLAY
3 SEARCH
4 SORT
enter your choice:
4

the sorted list of names are:
Employee Details:
Employee-Id:2
Name:albin
Salary:25000.000000
Employee-Id:1
Name:karthik
Salary:29000.000000

Do you want to continue (1/0)

PROGRAM NO: 27

AIM: Program to implement union.

ALGORITHM

Step 1: start

Step 2: define union student with rollno,name and mark

Step 3: create a variable of student union

Step 4: call function stud

Step 5: stop

Function stud()

Step 1: start

Step 2: read rollno

Step 3: display rollno

Step 4: read name

Step 5: display name

Step 6: read mark

Step 7: display mark

Step 8: stop

PROGRAM

```
union student
{
int roll_no;
char name[20];
int mark;
}stu;
void stud();
```

```
main()
{
stud();
}
void stud()
{
printf("\nenter the roll no");
scanf("%d",&stu.roll_no);
printf("\nthe student roll number is:%d",stu.roll_no);
printf("\nenter the student name");
scanf("%s",stu.name);
printf("\nthe student name is       :%s",stu.name);
printf("\nenter the student mark");
scanf("%d",&stu.mark);
printf("\nthe student mark is       :%d",stu.mark);
}
```

OUTPUT:

enter the roll no

4

the student roll number is:4

enter the student name

albin

the student name is :albin

enter the student mark

40

the student mark is :40

PROGRAM NO: 28

AIM: Telephone directory using file.

ALGORITHM

Step 1 : start

Step 2 : create a structure telephone with id,name,area,phno

Step 3 : create a structure variable t

Step 4 : print menu and select choice ch

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

and go to step 12

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

and go to step 12

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

and go to step 12

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

and go to step 12

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

and go to step 12

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

Step 11 : print “exited”

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

Step 13 : stop

Function insert()

Step 1 : start

Step 2 : declare the file pointer fp

Step 3 : open the file with a+ mode

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

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

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

Step 7 : close the file

Step 8 : end

Step 9 : stop

Function view()

Step 1 : start

Step 2 : declare the file pointer fp

Step 3 : open a file with r mode

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

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

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

go to step 8

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

go to step 6

Step 8 : close the file

Step 9 : stoP

Function search()

Step 1 : start

Step 2 : declare file pointer fp

Step 3 : open a file with r mode

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

Step 5 : Print “No records”

Step 6 : read the value of n from user

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

go to step 11

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

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

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

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

Step 12 : print “ can not found”

Step 13 : close the file

Step 14 : stop

Function update()

Step 1 : start

Step 2 : declare the file pointer fp

Step 3 : set f equal to 0

Step 4 : open a file with r+ mode

Step 5 : set the pointer fp at first

Step 6 : set r equal to size of the file

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

Step 8 : Print ”No records”

Step 9 : read the value of n from user

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

otherwise go to step 15

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

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

to the file

Step 13 : Set the file pointer to the particular point

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

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

Step 16 : print “ can not found”

Step 17 : close the file

Step 18 : stop

Function delete()

Step 1 : start

Step 2 : declare file pointers fp and f1

Step 3 : set f equal to 0

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

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

Step 6 : Print “No records”

Step 7 : read the value of n from user

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

go to step 11

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

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

Step 11 : close the file fp

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

Step 13 : close the file f1

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

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

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);
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:
albin
```

Enter the area:

peravoor

Enter the phone number:9744500406

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

1

Enter the Id:

2

Enter the name:

ashin

Enter the area:

mattanur

Enter the phone number:9497487886

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

2

ID	NAME	AREA	Ph Number
1	albin	peravoor	9744500406
2	ashin	mattanur	9497487886

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 albin peravoor 9744500406

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

3

Enter one ID you want to search:

5

Cannot found

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:

3

Enter the new name:

appachan

Enter the new area:

manathana

Enter the new phone number:

9388692020

Suucefully updated

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

2

ID	NAME	AREA	Ph Number
3	appachan	manathana	9388692020
2	ashin	mattanur	9497487886

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 NAMEAREA Ph Number

2 ashin mattanur 9497487886

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

5

Enter the ID to be deletd:

6

Id not found

1.INSERT

2.VIEW

3.SEARCH

4.UPDATE

5.DELETE

6.EXIT

Enter the option:

PROGRAM NO: 29

AIM: Program to update record in a file.

ALGORITHM

Step1: start

Step 2: create a structure emp

Step 3: create a structure variable e

Step 4: declare the file pointer fp

Step 5: open the file with w mode

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

Step 7: close the file

Step 8: open the file with r mode

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

Step10: exit

Step 11: read the position to be updated

Step 12: write new data

Step 13: close the file

Step 14: open the file with r mode

Step 15:read the data from the file

Step 16:stop

PROGRAM:

```
#include<stdio.h>

#include<stdlib.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
3
Enter Emp No,Emp Name and Salary 3 Employees
1 swetha 25000
2 Albin 30000
3 ashin 40000
Contents of the file emp.txt
1       swetha 25000.000000
2       Albin   30000.000000
3       Ashin   40000.000000
Enter Record No of the record to be updated
2
Enter new details of the employee
2 Albi 20000
Contents of the file emp.txt
1       swetha 25000.000000
2       Albin   20000.000000
3       ashin    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 : if i less than argc go to step 7 otherwise go to step 9

Step 7 : print the value argv[i]

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

Step 9 : close the file

Step 10 : print the value of argv[1]

Step 11 : open the file with r mode

Step 12 : set i equal to 2

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

```
[albi@DBCPC1 Desktop]$ ./a.out albin.txt welcome to kerala

Number of arguments 5

Content of albin.txt file

welcometokerala
```

DATABASE MANAGEMENT SYSTEM

USING POSTGRESQL

1. DDL AND DML COMMANDS

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

```
albin=# 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.

```
albin=# insert into book values(101,'nalukett','mtvasudevan');
INSERT 0 1
albin=# insert into book values(102,'my story','kamala das');
INSERT 0 1
albin=# insert into book values(103,'balyakalasakhi','basheer');
INSERT 0  1
```

3. Show the contents of the table.

```
albin=# select * from book;
 id |       name         | author
-----+----------------- +-----------
101 | nalukett          | mtvasudevan
102 |  my story           | kamala das
103 | balyakalasakhi    | basheer
```

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

```
albin=# alter table book add price int;
ALTER TABLE
```

5. Show the contents of the table.

albin=# select * from book;

id | name | author | price

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

101 | nalukett | mtvasudevan |

102 | my story | kamala das |

103 | balyakalasakhi | basheer |

(3 rows)

6. Update the price of each book

albin=# update book set price=200 where id=101;

UPDATE 1

albin=# update book set price=250 where id=102;

UPDATE 1

albin=# update book set price=150 where id=103;

UPDATE 1

7. Show the content of the table.

albin=# select * from book;

id | name | author | price

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

101 | nalukett | mtvasudevan | 200

102 | my story | kamala das | 250

103 | balyakalasakhi | basheer | 150

(3 rows)

8.Alter the table by deleting the column author.

albin=# alter table book drop author;

ALTER TABLE

9.Show the contents of the table.

```
albin=# select * from book;
id    |       name             |   price
-----+-------------------+---------
101 | nalukett               |  200
102 |  my story           |   250
103 | balyakalasakhi   | 150
(3 rows)
```

10. Delete some rows from book.

```
albin=# delete from book where id=102;
DELETE 1
```

11. Show the content of the table.

```
albin=# select * from book;
id    |  name             |   price
-----+-------------------+------
101 | nalukett               |  200
103 | balyakalasakhi   | 150
(2 rows)
```

12. Drop the book table.

```
albin=# drop table book;
DROP TABLE
```

2. INTEGRITY CONSTRAINTS

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

```
albin=# 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

```
albin=# insert into depp values(101,'m ca',1);
INSERT 0 1
albin=# insert into depp values(102,'bba',1);
INSERT 0 1
albin=# insert into depp values(103,'bcom',2);
INSERT 0 1
albin=# insert into depp values(104,'english',3);
INSERT 0 1
albin=# insert into depp values(105,'physics',3);
INSERT 0 1
```

3. Show the contents of department table.

```
albin=# select * from depp;
 dno |   dname   | location
-----+-----------+---------
 101 | mca       |        1
 102 | bba   |       1
 103 | bcom     |        2
 104 | english   |        3
 105 | physics |        3
(5 rows)
```

4. Create table employee with fields employe_id number,name string,address string,salary number,dpartment_no number.

albin=# 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.

albin=# insert into employ values(201,'albin','kunnnumpurath',10000,101);

INSERT 0 1

albin=# insert into employ values(202,'sreeedar','odakkal',16000,102);

INSERT 0 1

albin=# insert into employ values(203,'arunn','illikal',20000,102);

INSERT 0 1

albin=# insert into employ values(204,'athul','manasa',18000,103);

INSERT 0 1

albin=# insert into employ values(205,'maniyan','arumakkan',18000,101);

INSERT 0 1

6. Show the contents of employee table.

albin=# select * from employ;

```
 eid |  ename   | address  | salary | dno
-----+---------+---------+-------+----
 201 | albin    | kunnnumpurath |  10000 | 101
 202 | ashinn| odakkal|  16000 | 102
 203 | arunn  | illikal|  20000 | 102
 204 | athul   | manasa |  18000 | 103
 205 | maniyan    | arumakkan|  18000 | 101
(5 rows)
```

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

```
albin=# 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.

```
albin=#  select e.ename,d.dname from depp d,employ e where d.dno=e.dno;
  ename   |  dname
----------+--------
 albin    | mca
 ashinn| bba
 arunn  | bba
 athul   | bcom
 maniyan    | mca
(5 rows)
```

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

```
albin=#  select ename,salary from employ where salary=(select max(salary) from
employ);
  ename  | salary
---------+-------
 arunn |  20000
(1 row)
```

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

```
albin=# select ename from employ where salary between 10000 and 20000;
  ename
----------
 albin
```

ashinn

arunn

athul

maniyan

(5 rows)

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

albin=# 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

---

maniyan

(1 row

3. AGGREGATE FUNCTION

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

albin=# 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.

albin=# insert into foodcart values(200,'chicken','10-12-2018',40);
INSERT 0 1
albin=# insert into foodcart values(201,'beef','11-13-2048',60);
INSERT 0 1
albin=# insert into foodcart values(202,'pork','01-14-2023',100);
INSERT 0 1
albin=# insert into foodcart values(203,'mutton','09-28-2021',78);
INSERT 0 1
albin=# insert into foodcart values(204,'butter','02-25-2017',55);
INSERT 0 1
albin=# insert into foodcart values(205,'chees','06-23-2012',44);
INSERT 0 1

3.Show the contents of foodcart.

albin=# select * from foodcart;

```
 fid |  fname  |    date    | sold
-----+--------+-----------+-----
 200 | chicken | 2018-10-12 |   40
  201 | beef    | 2048-11-13 |   60
```

```
 202 | pork    | 2023-01-14 |  100
 203 | mutton  | 2021-09-28 |   78
 204 | butter  | 2017-02-25 |   55
 205 | chees   | 2012-06-23 |   44
(6 rows)
```

4. Alter the table by adding field price.

```
albin=# alter table foodcart add price int;
ALTER TABLE
```

5. Show the contents of foodcart.

```
albin=# select * from foodcart;
 fid |  fname  |    date    | sold | price
-----+---------+------------+------+------
 200 | chicken | 2018-10-12 |   40 |
 201 | beef    | 2048-11-13 |   60 |
 202 | pork    | 2023-01-14 |  100 |
 203 | mutton  | 2021-09-28 |   78 |
 204 | butter  | 2017-02-25 |   55 |
 205 | chees   | 2012-06-23 |   44 |
(6 rows)
```

6. Insert values to price field.

```
albin=# select * from foodcart;
 fid |  fname  |    date    | sold | price
-----+---------+------------+------+------
 200 | chicken | 2018-10-12 |   40 |
 201 | beef    | 2048-11-13 |   60 |
 202 | pork    | 2023-01-14 |  100 |
 203 | mutton  | 2021-09-28 |   78 |
 204 | butter  | 2017-02-25 |   55 |
 205 | chees   | 2012-06-23 |   44 |
(6 rows)
albin=# update foodcart set price=60 where fid=200;
```

```
UPDATE 1
albin=# update foodcart set price=68 where fid=201;
UPDATE 1
albin=# update foodcart set price=79 where fid=202;
UPDATE 1
albin=# update foodcart set price=89 where fid=203;
UPDATE 1
albin=# update foodcart set price=200 where fid=204;
UPDATE 1
albin=# update foodcart set price=456 where fid=205;
UPDATE 1
```

7. Show the contents of foodcart table.

```
albin=# select * from foodcart;
 fid |  fname  |    date    | sold | price
-----+---------+------------+------+-------
 200 | chicken | 2018-10-12 |   40 |    60
 201 | beef    | 2048-11-13 |   60 |    68
 202 | pork    | 2023-01-14 |  100 |    79
 203 | mutton  | 2021-09-28 |   78 |    89
 204 | butter  | 2017-02-25 |   55 |   200
 205 | chees   | 2012-06-23 |   44 |   456
(6 rows)
```

8. Delete one row from foodcart.

```
albin=# delete from foodcart where fid=200;
DELETE 1
```

9. Show the conents of foodcart.

```
albin=# select * from foodcart;
 fid | fname  |    date    | sold | price
-----+--------+------------+------+-------
 201 | beef   | 2048-11-13 |   60 |    68
```

```
 202 | pork    | 2023-01-14 |  100 |    79
 203 | mutton | 2021-09-28 |   78 |    89
 204 | butter | 2017-02-25 |   55 |   200
 205 | chees  | 2012-06-23 |   44 |   456
(5 rows)
```

10. Retrieve number of items in foodcart.

```
albin=# select count(*) from foodcart;
 count
-------
     5
(1 row)
```

11. Retrieve sum of sold in food cart.

```
albin=# select sum(sold) from foodcart;
 sum
-----
 337
(1 row)
```

12. Retrieve average price from foodcart.

```
albin=# select avg(price) from foodcart;
         avg
----------------------
 178.4000000000000000
(1 row)
```

13. Retrieve   maximum price from foodcart.

```
albin=# select max(price) from foodcart;
 max
-----
 456
(1 row)
```

14. Retrieve minimum sold from foodcart.

```
albin=# select min(sold) from foodcart;
 min
-----
  44
(1 row)
```

15. Insert values into foodcart.

```
albin=# insert into foodcart values(207,'chicken','10-24-2026',70);
INSERT 0 1
albin=# insert into foodcart values(208,'chicken','10-24-2026',70);
INSERT 0 1
```

16. Show the contents of foodcart.

```
albin=# select * from foodcart;
 fid |   fname   |    date    | sold | price
-----+----------+-----------+-----+------
 201 | beef      | 2048-11-13 |   60 |    68
 202 | pork      | 2023-01-14 |  100 |    79
 203 | mutton    | 2021-09-28 |   78 |    89
 204 | butter    | 2017-02-25 |   55 |   200
 205 | chees     | 2012-06-23 |   44 |   456
 206 | sandwhich | 2026-10-24 |   70 |
 207 | chicken   | 2026-10-24 |   70 |
 208 | chicken   | 2026-10-24 |   70 |
(8 rows)
```

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

```
albin=# select fname,sum(sold) as totalsold from foodcart group by fname;
  fname   | totalsold
----------+----------
 chicken  |       140
 beef     |        60
```

```
 chees     |       44
 sandwhich |       70
 butter    |       55
 mutton    |       78
 pork      |      100
(7 rows)
```

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

albin=# select fname , avg(price) from foodcart group by fname having avg(price) >20;

```
 fname  |        avg
--------+--------------------
 beef   | 68.0000000000000000
 chees  | 456.0000000000000000
 butter | 200.0000000000000000
 mutton |  89.0000000000000000
 pork   |  79.0000000000000000
(5 rows)
```

4. SET OPERATIONS

1. Create table depositor.

albin=# create table depositor(id int primary key,name varchar(20),amount int,branch varchar(20));
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "depositor_pkey" for table "depositor"
CREATE TABLE

2. Insert values into depositor.

albin=# insert into depositor values(1,'ashin',4000,'kannur');
INSERT 0 1
albin=# insert into depositor values(2,'amrutha',45000,'kozhikode');
INSERT 0 1
albin=# insert into depositor values(3,'abin',45500,'kochi');
INSERT 0 1
albin=# insert into depositor values(4,'ammu',5600,'trivandrum');
INSERT 0 1
albin=# insert into depositor values(5,'rose',38000,'kollam');
INSERT 0 1

3.Show the contents of depositor.

albin=# select * from depositor;
 id | name | amount | branch
----+---------+--------+------------
  1 | ashin | 4000 | kannur
  2 | amrutha | 45000 | kozhikode
  3 | abin | 45500 | kochi
  4 | ammu | 5600 | trivandrum
  5 | rose | 38000 | kollam
(5 rows)

4.Create table borrower.

albin=# create table borower(id int primary key,name varchar(20),amout int,branch varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "borower_pkey" for table "borower"

CREATE TABLE

5. Insert values into borrower.

albin=# insert into borower values(101,'arun',46000,'peravoor');

INSERT 0 1

albin=# insert into borower values(102,'jose',76899,'itrity');

INSERT 0 1

albin=# insert into borower values(103,'rajan',75600,'manathana');

INSERT 0 1

albin=# insert into borower values(104,'stevo',65999,'kelakam');

INSERT 0 1

albin=# insert into borower values(105,'sudin',23900,'edathotty');

INSERT 0 1

albin=# insert into borower values(106,'ashin',23900,'mattanur');

INSERT 0 1

6. Show the contents of borrower.

albin=# select * from borower;

```
 id  | name  | amout |  branch
-----+-------+-------+-----------
 101 | arun  | 46000 | peravoor
 102 | jose  | 76899 | itrity
 103 | rajan | 75600 | manathana
 104 | stevo | 65999 | kelakam
 105 | sudin | 23900 | edathotty
```

106 | ashin | 23900 | mattannur

(5 rows)

7. Perform union operation with depositor and borrower table.

albin=# (select name from borower)union(select name from depositor);

```
 name
---------
 sudin
 rajan
 ammu
 arun
 jose
 stevo
 rose
 amrutha
 abin
 ashin
(10 rows)
```

8. Perform intersect operation with depositor and borrower table.

albin=# (select name from borower)intersect(select name from depositor);

```
 name
-------
 ashin
(1 row)
```

9. Perform except operation with depositor and borrower table.

albin=# (select name from borower)except(select name from depositor);

```
 name
-------
 stevo
 arun
 sudin
```

rajan

jose

(5 rows)

5. VIEW

1. Create a table project with fields project_id,project_name,and hours.

albin=# create table project(pid int primary key,pname varchar(20),hours int);

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

CREATE TABLE

2. Insert tuples into project table.

albin=# insert into project values(101,'dbms',2);

INSERT 0 1

albin=# insert into project values(102,'os',8);

INSERT 0 1

albin=# insert into project values(103,'ds',5);

INSERT 0 1

albin=# insert into project values(104,'mathematics',2);

INSERT 0 1

albin=# insert into project values(105,'prog',6);

INSERT 0 1

3. Show the contents of project.

albin=# select * from project;

pid | pname | hours

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

101 | dbms | 2

102 | os | 8

103 | ds | 5

104 | mathematics | 2

105 | prog | 6

(5 rows)

4. Create a table employee

albin=# create table employ(eid int primary key,fname varchar(20),lname varchar(20),pid int references project);

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

CREATE TABLE

5. Insert values into employee table.

albin=# insert into employ values(1,'anu','akhil',101);

INSERT 0 1

albin=# insert into employ values(2,'anagha','athira',101);

INSERT 0 1

albin=# insert into employ values(3,'albin','ashin',103);

INSERT 0 1

albin=# insert into employ values(4,'dias','ebin',104);

INSERT 0 1

albin=# insert into employ values(5,'dileep','ebin',104);

INSERT 0 1

albin=# insert into employ values(6,'jes','jis',105);

INSERT 0 1

6. Show the contents of employee table.

albin=# select * from employ;

```
 eid | fname  | lname  | pid
-----+--------+--------+-----
   1 | anu    | akhil  | 101
   2 | anagha | athira | 101
   3 | albin  | ashin  | 103
   4 | dias   | ebin   | 104
   5 | dileep | ebin   | 104
   6 | jes    | jis    | 105
(6 rows)
```

7. Create a view.

albin=# create view prjemp as select fname,lname,pname,hours from employ,project where employ.pid=project.pid;

CREATE VIEW

8. Show the contents of new view.

albin=# select * from prjemp;

```
 fname  | lname |    pname    | hours
--------+-------+-------------+-------
 anu    | akhil | dbms        |     2
 anagha | athira | dbms       |     2
```

albin | ashin | ds | 5

dias | ebin | mathematics | 2

dileep | ebin | mathematics | 2

jes | jis | prog | 6

(6 rows)

9. Select value from view where project name is equal to 'ds'

albin=# select * from prjemp where pname='ds';

fname | lname | pname | hours

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

albin | ashin | ds | 5

(1 row)

10. Select value from view where project name is equal to 'dbms'

albin=# select * from prjemp where pname='dbms';

fname | lname | pname | hours

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

anu | akhil | dbms | 2

anagha | athira | dbms | 2

(2 rows)

11. Select value from view where hours greater than 4 and less than 4.

albin=# select * from prjemp where hours > 4;

fname | lname | pname | hours

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

```
 albin | ashin | ds    |     5

 jes   | jis   | prog  |     6

(2 rows)

albin=# select * from prjemp where hours < 4;

 fname  | lname  |    pname    | hours

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

 anu    | akhil  | dbms        |     2

 anagha | athira | dbms        |     2

 dias   | ebin   | mathematics |     2

 dileep | ebin   | mathematics |     2

(4 rows)
```

6. FUNCTION

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

1. Create table student.

albin=# create table student(id int primary key,name varchar(20),mark float);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "student_pkey" for table "student"

CREATE TABLE

2.Insert values into student table.

albin=# insert into student values(101,'ashin',60);

INSERT 0 1

albin=# insert into student values(102,'ebin',50);

INSERT 0 1

albin=# insert into student values(103,'albin',70);

INSERT 0 1

albin=# insert into student values(104,'dias',33.55);

INSERT 0 1

albin=# insert into student values(105,'dileep',66.5);

INSERT 0 1

3. Show the contents of student table.

albin=# select * from student;

id | name | mark

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

101 | ashin | 60

102 | ebin | 50

103 | albin | 70

104 | dias | 33.55

105 | dileep | 66.5

(5 rows)

4. Create fuction.

albin=# create function plpgsql_call_handler() returns opaque as '$libdir/plpgsql.so' language'c';

CREATE FUNCTION

5. Create language.

albin=# create language 'plpgsql' handler plpgsql_call_handler lancompiler 'pl/pgsql';

NOTICE: using pg_pltemplate information instead of CREATE LANGUAGE parameters

CREATE LANGUAGE

albin=# create function funalbin(text) returns integer as'

declare

a int;

begin

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

return a;

end 'language'plpgsql';

bash-4.1$ psql albin
albin=# \i fun.sql

CREATE FUNCTION

6. Show the contents of student table.

```
albin=# select * from student;
 id  |  name  | mark
-----+--------+-------
 101 | ashin  |    60
 102 | ebin   |    50
 103 | albin  |    70
 104 | dias   | 33.55
 105 | dileep |  66.5
(5 rows)
albin=# select funalbin('albin');
 funalbin
----------
       70
(1 row)
albin=# select funalbin('ebin');
 funalbin
----------
       50
(1 row)
albin=# select funalbi('dileep');
 funalbi
```

66.5

(1 row)

II. Create a function to add two numbers.

```
create function sum(a int,b int)returns integer as'
begin
return a+b;
end
'language'plpgsql';

bash-4.1$ psql albin
albin=# \i funsum2.sql
CREATE FUNCTION
```

Find Sum.

```
albin=# select sum(3,2);
 sum
-----
   5
(1 row)

albin=# select sum(5,5);
 sum
-----
  10
(1 row)
```

7. TRIGGER

1. Create table student.

albin=# create table student(id int primary key,name varchar(20),mark float);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "student_pkey" for table "student"

CREATE TABLE

2.Insert values into student table.

albin=# insert into student values(101,'ashin',60);

INSERT 0 1

albin=# insert into student values(102,'ebin',50);

INSERT 0 1

albin=# insert into student values(103,'albin',70);

INSERT 0 1

albin=# insert into student values(104,'dias',33.55);

INSERT 0 1

albin=# insert into student values(105,'dileep',66.5);

INSERT 0 1

3. Show the contents of student table.

albin=# select * from student;

 id | name | mark

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

 101 | ashin | 60

102 | ebin | 50

103 | albin | 70

104 | dias | 33.55

105 | dileep | 66.5

(5 rows)

4. Create a trigger.

```
create function fun() returns trigger as'
begin
update stud set mark=mark+30;
return new;
end
'language'plpgsql';
```

```
-bash-4.1$ psql albin
albin=# \i funtrg.sql;
CREATE FUNCTION
```

```
albin=# create trigger trig after insert on student execute procedure fun();
CREATE TRIGGER
```

5. Show the contents of student table.

```
albin=# select * from student;
```

id | name | mark

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

101 | ashin | 60

102 | ebin | 50

103 | albin | 70

104 | dias | 33.55

105 | dileep | 66.5

(5 rows)

6. Insert a new value to student table.

albin=# insert into student values(106,'arun',5699);

INSERT 0 1

7. Show the contents of student table.

albin=# select * from student;

id | name | mark

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

101 | ashin | 110

102 | ebin | 100

103 | albin | 120

104 | dias | 83.55

105 | dileep | 116.5

106 | arun | 5749

(6 rows)

8. MATHEMATICAL FUNCTIONS

1. sin()

```
albin=# select sin(90);
        sin
-------------------
 0.893996663600558
(1 row)
```

2. cos()

```
albin=# select cos(0);
 cos
-----
   1
(1 row)
```

3. tan()

```
albin=# select tan(45);
       tan
------------------
 1.61977519054386
(1 row)
```

4. sqrt()

```
albin=# select sqrt(25);
 sqrt
------
    5
```

(1 row)

5. pow()

albin=# select pow(2,5);

 pow

  32

(1 row)

6. pi()

albin=# select pi();

        pi

 3.14159265358979

(1 row)

7. radians()

albin=# select radians(90);

     radians

 1.5707963267949

(1 row)

8. log()

albin=# select log(180);

       log

 2.25527250510331

(1 row)

9. abs()

albin=# select abs(-19);

 abs

  19

(1 row)

10. atan()

albin=# select atan(20);

       atan

 1.52083793107295

(1 row)

11. acos()

albin=# select acos(1);

 acos

    0

(1 row)

12. trunc()

albin=# select trunc(17.54);

 trunc

    17

(1 row)

13.round()

albin=# select round(34.88);

round

35

(1 row)

14.round()

albin=# select round(34.22);

round

34

(1 row)

15.random()

albin=# select round(34.45362645);

round

34

(1 row)

16.exp()

albin=# select exp(20);

exp

485165195.40979

(1 row)

```
albin=# select exp(30);
       exp
------------------
 10686474581524.5
(1 row)
```

17.floor()

```
albin=# select floor(1.20);
 floor
-------
     1
(1 row)
albin=# select floor(1.90);
 floor
-------
     1
(1 row)
```

18.cbrt()

```
albin=# select cbrt(9);
      cbrt
-----------------
 2.0800838230519
(1 row)
```

19.ln()

```
albin=# select ln(28) as natural_log;

   natural_log

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

 3.3322045101752

(1 row)
```

20.ceil

```
albin=# select ceil(3.8);

 ceil

------

    4

(1 row)
```

9. CHARACTER STRING FUNCTIONS

```
albin=# select ascii('T');
 ascii
-------
    84
(1 row)
albin=# select ascii('P');
 ascii
-------
    80
(1 row)
albin=# select btrim('hello world','world');
 btrim
--------
 hello


 (1 row)
albin=# select ltrim('       thomas');
 ltrim
--------
 thomas
(1 row)
albin=# select rtrim('       albin');
```

```
   rtrim
--------------
       albin
(1 row)
albin=# select initcap('arun');
 initcap
---------
 Arun
(1 row)

albin=# select length('arun');
 length
--------
      4
(1 row)
albin=# select substr('welcome',3);
 substr
--------
 lcome
(1 row)
albin=# select substr('welcome',2,3);
 substr
--------
 elc
```

```
(1 row)

albin=# select substr('helloworld',6,5);

 substr

--------

 world

(1 row)

albin=# select lower('MATEW');

 lower

-------

 matew

(1 row)

albin=# select upper('gangan');

 upper

--------

 GANGAN

(1 row)

albin=# select chr('81');

 chr

-----

 Q

(1 row)

albin=# select translate('albin joseph','joseph','thomas');

  translate

--------------
```

albin thomas

(1 row)

10. TRANSACTION

1. Show the contents from student table.

```
albin=# select * from student;
 id |   name   | mark |   place
-----+----------+------+------------
 102 | ebin     |  200 | iritty
 103 | albin    |  220 | kannur
 101 | adithyan |  210 | peravoor
 104 | ashin    |  300 | thalassery
 105 | arun     |  310 | kozhikode
(5 rows)
albin=# begin;
BEGIN
albin=# update student set name='aaruni' where id=102;
UPDATE 1
albin=# update student set mark=400  where id=104;
UPDATE 1
albin=# select * from student;
 id |   name   | mark |   place
-----+----------+------+------------
 103 | albin    |  220 | kannur
 101 | adithyan |  210 | peravoor
 105 | arun     |  310 | kozhikode
 102 | aaruni   |  200 | iritty
 104 | ashin    |  400 | thalassery
(5 rows)

albin=# rollback;
ROLLBACK
albin=# select * from student;
```

```
 id |   name   | mark |   place
-----+----------+------+------------
 102 | ebin     |  200 | iritty
 103 | albin    |  220 | kannur
 101 | adithyan |  210 | peravoor
 104 | ashin    |  300 | thalassery
 105 | arun     |  310 | kozhikode
(5 rows)

albin=# begin;
BEGIN
albin=# delete from student where id=105;
DELETE 1
albin=# update student set name='amrutha' where id=102;
UPDATE 1
albin=# commit;
COMMIT
albin=# select * from student;
 id |   name   | mark |   place
-----+----------+------+------------
 103 | albin    |  220 | kannur
 101 | adithyan |  210 | peravoor
 104 | ashin    |  300 | thalassery
 102 | amrutha  |  200 | iritty
(4 rows)
```

11. CURSOR

1. Show the contents of student table.

```
albin=# select * from student;

 id |   name   | mark |   place

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

 103 | albin    |  220 | kannur

 101 | adithyan |  210 | peravoor

 104 | ashin    |  300 | thalassery

 102 | amrutha  |  200 | iritty

(4 rows)


albin=# begin;

BEGIN


albin=# declare firstcursor cursor for select name from student;

DECLARE CURSOR


albin=# fetch 3 from firstcursor;

   name

----------

 albin

 adithyan

 ashin
```

```
(3 rows)

albin=# fetch 2 from firstcursor;
  name
---------
 amrutha
(1 row)

albin=# fetch 1 from firstcursor;
 name
------
(0 rows)

albin=# move backward 4 in firstcursor;
MOVE 4
albin=# fetch 1 from firstcursor;
   name
----------
 adithyan
(1 row)

albin=# fetch 3 from firstcursor;
  name
---------
```

```
 ashin
 amrutha
(2 rows)

albin=# move backward 5 in firstcursor;
MOVE 4
albin=# fetch 3 from firstcursor;
   name
----------
 albin
 adithyan
 ashin
(3 rows)

albin=# fetch next from firstcursor;
  name
---------
 amrutha
(1 row)

albin=# fetch prior from firstcursor;
 name
-------
 ashin
```

```
(1 row)

albin=# fetch prior from firstcursor;
  name
----------
 adithyan
(1 row)

albin=# move forward 1 in firstcursor;
MOVE 1
albin=# fetch prior from firstcursor;
  name
----------
 adithyan
(1 row)

albin=# fetch next from firstcursor;
 name
-------
 ashin
(1 row)

albin=# move backward 5 in firstcursor;
MOVE 2
```

```
albin=# fetch next from firstcursor;
 name
-------
 albin
(1 row)

albin=# fetch 1 from firstcursor;
   name
----------
 adithyan
(1 row)

albin=# fetch 5 from firstcursor;
  name
---------
 ashin
 amrutha
(2 rows)
```

12. INDEX

```
albin=# create index firtindex on student(name);

CREATE INDEX

albin=# \d student;

          Table "public.student"

 Column |         Type          | Modifiers

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

 id     | integer               | not null

 name   | character varying(20) |

 mark   | double precision      |

 place  | character varying(20) |

Indexes:

    "student_pkey" PRIMARY KEY, btree (id)

    "firtindex" btree (name)

Triggers:

    trig AFTER INSERT ON student FOR EACH STATEMENT EXECUTE
PROCEDURE fun()
```

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.

albin=# create table customer(custno int primary key,cname varchar(10),city varchar(10));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "customer_pkey" for table "customer"

CREATE TABLE

albin=# create table order2(orderno int primary key,odate date,ord_amt int,custno int references customer);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "order2_pkey" for table "order2"

CREATE TABLE

albin=# 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

albin=# create table order_item(orderno int,itemno int,quantity int,primary key(orderno,itemno),foreign key(orderno) references order2(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

albin=# 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

albin=# create table shipment(orderno int,warehouseno int,ship_date date,primary key(orderno,warehouseno),foreign key(orderno) references order2(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.

albin=# insert into customer values(100,'aranya','arabi');
INSERT 0 1
albin=# insert into customer values(102,'swetha','kochi');
INSERT 0 1
albin=# insert into customer values(101,'ashin','ernakulam');
INSERT 0 1
albin=# insert into customer values(103,'albin','kannur');
INSERT 0 1
albin=# insert into customer values(104,'anagha','kozhikode');
INSERT 0 1
albin=# select * from customer;

```
 custno | cname  |   city
--------+--------+-----------
    100 | aranya | arabi
    102 | swetha | kochi
    101 | ashin  | ernakulam
    103 | albin  | kannur
    104 | anagha | kozhikode
(5 rows)

albin=# insert into order2 values(200,'04-12-2019',2700,100);
INSERT 0 1
albin=# insert into order2 values(201,'11-11-2017',4500,101);
INSERT 0 1
albin=# insert into order2 values(202,'09-18-2013',8900,102);
INSERT 0 1
albin=# insert into order2 values(203,'08-28-2012',87800,103);
INSERT 0 1
albin=# insert into order2 values(204,'01-22-2011',2300,104);
INSERT 0 1
albin=# select * from order2;
 orderno |   odate    | ord_amt | custno
---------+------------+---------+--------
     200 | 2019-04-12 |    2700 |    100
     201 | 2017-11-11 |    4500 |    101
     202 | 2013-09-18 |    8900 |    102
     203 | 2012-08-28 |   87800 |    103
     204 | 2011-01-22 |    2300 |    104
(5 rows)

albin=# insert into item values(300,47000);
INSERT 0 1
albin=# insert into item values(301,6000);
INSERT 0 1
albin=# insert into item values(302,2300);
```

```
INSERT 0 1
albin=# insert into item values(303,4500);
INSERT 0 1
albin=# insert into item values(304,11222);
INSERT 0 1
albin=# select * from item;
 itemno | unitprice
--------+-----------
    300 |     47000
    301 |      6000
    302 |      2300
    303 |      4500
    304 |     11222
(5 rows)

albin=# insert into order_item values(200,300,34000);
INSERT 0 1
albin=# insert into order_item values(201,300,4500);
INSERT 0 1
albin=# insert into order_item values(201,301,4500);
INSERT 0 1
albin=# insert into order_item values(200,301,67000);
INSERT 0 1
albin=# insert into order_item values(200,302,45677);
INSERT 0 1
albin=# insert into order_item values(201,302,5600);
INSERT 0 1
albin=# insert into order_item values(203,303,5600);
INSERT 0 1
albin=# select * from order_item;
 orderno | itemno | quantity
---------+--------+----------
     200 |    300 |    34000
     201 |    300 |     4500
```

```
   201 |   301 |    4500
   200 |   301 |   67000
   200 |   302 |   45677
   201 |   302 |    5600
   203 |   303 |    5600
(7 rows)

albin=# insert into warehouse values(500,'kannur');
INSERT 0 1
albin=# insert into warehouse values(501,'kochi');
INSERT 0 1
albin=# insert into warehouse values(502,'ernakulam');
INSERT 0 1
albin=# insert into warehouse values(503,'kollam');
INSERT 0 1
albin=# insert into warehouse values(504,'peravoor');
INSERT 0 1
albin=# select * from warehouse;
 warehouseno |   city
-------------+-----------
         500 | kannur
         501 | kochi
         502 | ernakulam
         503 | kollam
         504 | peravoor
(5 rows)
```

```
albin=# insert into shipment values(200,500,'10-11-2019');
INSERT 0 1
albin=# insert into shipment values(201,500,'11-20-2012');
INSERT 0 1
albin=# insert into shipment values(201,501,'01-30-2012');
INSERT 0 1
albin=# insert into shipment values(202,500,'02-29-2012');
INSERT 0 1
albin=# insert into shipment values(202,504,'05-19-2013');
INSERT 0 1
albin=# insert into shipment values(203,504,'11-26-2010');
INSERT 0 1
albin=# insert into shipment values(203,503,'06-25-2098');
INSERT 0 1
albin=# select * from shipment;
 orderno | warehouseno | ship_date
---------+-------------+------------
     200 |         500 | 2019-10-11
     201 |         500 | 2012-11-20
     201 |         501 | 2012-01-30
     202 |         500 | 2012-02-29
     202 |         504 | 2013-05-19
     203 |         504 | 2010-11-26
```

203 | 503 | 2098-06-25

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.

```
albin=# select customer.custno,count(*) as no_of_orders,avg(ord_amt) as
avg_order_amount from customer,order2 where customer.custno=order2.custno
group by customer.custno;
 custno | no_of_orders |   avg_order_amount
--------+--------------+----------------------
    101 |            1 | 4500.0000000000000000
    104 |            1 | 2300.0000000000000000
    102 |            1 | 8900.0000000000000000
    100 |            1 | 2700.0000000000000000
    103 |            1 |    87800.000000000000
(5 rows)
albin=# delete from item where itemno=300;
DELETE 1
albin=# select * from item;
 itemno | unitprice
--------+-----------
    301 |      6000
    302 |      2300
    303 |      4500
```

```
 304 |     11222
(4 rows)

albin=# select s.orderno from shipment s,warehouse w where
s.warehouseno=w.warehouseno;
 orderno
---------
     200
     201
     201
     202
     202
     203
     203
(7 rows)
```

14. EMPLOYEE DEPARTMENT RELATION

Consider the database given below. The primary keys are made bold and the data types are specified.

emp (**empno** number, ename varchar, job varchar, deptno number, sal number)
dept (**deptno** number, dname varchar, loc varchar)

1. Create the above tables by properly specifying the primary keys and foreign keys.

albin=# create table deptt(dno int not null primary key,dname varchar(15),location varchar(15));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "deptt_pkey" for table "deptt"

CREATE TABLE

albin=# create table empll(eno int not null primary key,ename varchar(15),job varchar(15),dno int references deptt,salary int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "empll_pkey" for table "empll"

CREATE TABLE

2. Enter at least five tuples for each relation.

albin=# insert into deptt values(101,'msw','ground_flr');
INSERT 0 1
albin=# insert into deptt values(102,'bba','ground_flr');
INSERT 0 1
albin=# insert into deptt values(103,'bsw','ground_flr');
INSERT 0 1
albin=# insert into deptt values(104,'ma','second_flr');

```
INSERT 0 1
albin=# insert into deptt values(105,'ba','third_flr');
INSERT 0 1

albin=# insert into empll values(16,'aruni','acountat',105,2000);
INSERT 0 1
albin=# insert into empll values(11,'arun','front',101,4550);
INSERT 0 1
albin=# insert into empll values(12,'ashion','keep',101,2000);
INSERT 0 1
albin=# insert into empll values(13,'ashin','kept',104,2900);
INSERT 0 1
```

3. Show the the contents of each relation.

```
albin=# select * from deptt;
 dno | dname |  location
-----+-------+-------------
 100 | mca   | second_flor
 101 | msw   | ground_flr
 102 | bba   | ground_flr
 103 | bsw   | ground_flr
 104 | ma    | second_flr
 105 | ba    | third_flr
(6 rows)

albin=# select * from empll;
 eno | ename  |   job    | dno | salary
-----+--------+----------+-----+--------
  16 | aruni  | acountat | 105 |   2000
  11 | arun   | front    | 101 |   4550
  12 | ashion | keep     | 101 |   2000
  13 | ashin  | kept     | 104 |   2900
(4 rows)
```

4. Retrieve name of the employee who get the highest salary from msw department.

```
albin=# select ename from empll where salary=(select max(salary) from empll,deptt
where empll.dno=deptt.dno and dname='msw');
 ename
-------
 arun
(1 row)
```

5. List the records in the emp table order by salary in ascending order.

```
albin=# select * from empll order by salary asc;
 eno | ename  |   job    | dno | salary
-----+--------+----------+-----+--------
  16 | aruni  | acountat | 105 |   2000
  12 | ashion | keep     | 101 |   2000
  13 | ashin  | kept     | 104 |   2900
  11 | arun   | front    | 101 |   4550
(4 rows)
```

6. Display deptno from the table employee avoiding the duplicated values.

```
albin=# select distinct dno from empll;
 dno
-----
 101
 105
 104
(3 rows)
```

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

```
albin=# select ename,dname from empll e,deptt d where e.dno=d.dno;
  ename  | dname
 --------+-------
```

```
 aruni  | ba
 arun   | msw
 ashion | msw
 ashin  | ma
(4 rows)
```

15. INSTRUCTOR-DEPARTMENT RELATION

1. Create department relation.

```
albin=# 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.

```
albin=# insert into dep values('bcom','albin','kannur');
INSERT 0 1
albin=# insert into dep values('bba','ashin','kozhikode');
INSERT 0 1
albin=# insert into dep values('ba','ebin','kollam');
INSERT 0 1
albin=# insert into dep values('ma','aswathi','trivandrum');
INSERT 0 1
albin=# insert into dep values('bsw','anupriya','kasargod');
INSERT 0 1
```

3. Show the contents of department relation.

```
 albin=# select * from dep;
 dep_name |   hod    |  location
----------+---------+-----------
 bcom     | albin    | kannur
 bba      | ashin    | kozhikode
 ba       | ebin     | kollam
 ma       | aswathi  | trivandrum
 bsw      | anupriya | kasargod
(5 rows)
```

4. Create an instructor relation.

albin=# 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.

albin=# insert into instruct values(100,'allesha','bcom',3000,'kannur');

INSERT 0 1

albin=# insert into instruct values(101,'kuttan','ba',56000,'kozhikode');

INSERT 0 1

albin=# insert into instruct values(102,'anu','ma',23999,'trivandrum');

INSERT 0 1

albin=# insert into instruct values(103,'amrutha','ba',45999,'peravoor');

INSERT 0 1

albin=# insert into instruct values(104,'anusree','bsw',3400,'iritty');

INSERT 0 1

6. Show the content of instructor table.

```
 albin=# select * from instruct;
 instrct_id |  name   | dep_name |  salary  |  address
------------+---------+----------+----------+-----------
        100 | allesha | bcom     |  3000.00 | kannur
        101 | kuttan  | ba       | 56000.00 | kozhikode
        102 | anu     | ma       | 23999.00 | trivandrum
        103 | amrutha | ba       | 45999.00 | peravoor
        104 | anusree | bsw      |  3400.00 | iritty
(5 rows)
```

7. Retrieve name and department name from instructor table.

```
albin=# SELECT name,dep_name from instruct;
 name   | dep_name
---------+---------
allesha | bcom
kuttan  | ba
anu     | ma
amrutha | ba
anusree | bsw
(5 rows)
```

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

```
albin=# SELECT name,dep_name from instruct where salary>20000.00;
 name   | dep_name
---------+---------
kuttan  | ba
anu     | ma
amrutha | ba
(3 rows)
```

9. Retrieve average salary from instructor.

```
albin=#  SELECT avg(salary) from instruct;
        avg
--------------------
 26479.600000000000
(1 row)
```

10. Select average salary from instructor and group by department name.

```
albin=# SELECT avg(salary) as avg_salary from instruct group by dep_name;

    avg_salary
----------------------
```

3400.0000000000000000

3000.0000000000000000

23999.000000000000

50999.500000000000

(4 rows)

11. Select name from instructor where salary between 10000 and 30000.

albin=# SELECT name from instruct where salary between 10000 and 30000;

name

---

anu

(1 row)

12. Delete one row from instructor where department name equal to 'ma'.

albin=# DELETE from instruct where dep_name='ma';

DELETE 1

13. Show the contents of instructor table.

albin=# select * from instruct;

instrct_id | name | dep_name | salary | address

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

100 | allesha | bcom | 3000.00 | kannur

101 | kuttan | ba | 56000.00 | kozhikode

103 | amrutha | ba | 45999.00 | peravoor

104 | anusree | bsw | 3400.00 | iritty

(4 rows)

14. Update instructor salary.

albin=# UPDATE instruct set salary=salary*1.05;

UPDATE 4

albin=# select * from instruct;

```
instrct_id |  name   | dep_name |  salary  |  address
------------+--------+---------+---------+----------
       100 | allesha | bcom     |  3150.00 | kannur
       101 | kuttan  | ba       | 58800.00 | kozhikode
       103 | amrutha | ba       | 48298.95 | peravoor
       104 | anusree | bsw      |  3570.00 | iritty
(4 rows)
```

16. 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.**

albin=# create table person(driver_id varchar(20) primary key,name varchar(20),address varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "person_pkey" for table "person"

CREATE TABLE

albin=# create table car(regno varchar(20) primary key,model varchar(20),year int);

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "car_pkey" for table "car"

CREATE TABLE

albin=# create table accident(report_no 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

albin=# create table owns(driver_id varchar(10) references person,regno varchar(20) references car);

CREATE TABLE

albin=# create table participated(driver_id varchar(20) references person,regno varchar(20) references car,report_number int references accident,damage_amout int);
CREATE TABLE

2. Enter tuples for each relation and show the contents

albin=# insert into person values('KL13','arun','odanthodu');

INSERT 0 1

albin=# insert into person values('KL12','rajan','manathana');

INSERT 0 1

albin=# insert into person values('KL14','nafseen','peravoor');

INSERT 0 1

albin=# insert into person values('KL15','nidhin','irity');

INSERT 0 1

albin=# insert into person values('KL16','ashin','kannur');

INSERT 0 1

albin=# select * from person;

```
 driver_id |  name   |  address
-----------+---------+-----------
 KL13      | arun    | odanthodu
 KL12      | rajan   | manathana
 KL14      | nafseen | peravoor
 KL15      | nidhin  | irity
 KL16      | ashin   | kannur
```

```
(5 rows)

albin=# insert into car values('KA24','BENZ',2029);
INSERT 0 1
albin=# insert into car values('KA21','MARUTHI',2018);
INSERT 0 1
albin=# insert into car values('KA54','TOYOTO',2016);
INSERT 0 1
albin=# insert into car values('KA34','BMW',3233);
INSERT 0 1
albin=# insert into car values('KA87','HYNDAI',7864);
INSERT 0 1
albin=# SELECT * FROM CAR;
 regno |  model  | year
-------+---------+------
 KA24  | BENZ    | 2029
 KA21  | MARUTHI | 2018
 KA54  | TOYOTO  | 2016
 KA34  | BMW     | 3233
 KA87  | HYNDAI  | 7864
(5 rows)

albin=# insert into accident values(450,'3/4/2019','kannur');
INSERT 0 1
```

```
albin=# insert into accident values(452,'6/7/2016','kozhikode');
INSERT 0 1
albin=# insert into accident values(454,'1/3/2015','kochi');
INSERT 0 1
albin=# insert into accident values(453,'2/3/2014','trivamndrum');
INSERT 0 1
albin=# insert into accident values(457,'10/9/2017','kollam');
INSERT 0 1
albin=# select * from accident;
 report_no |  acc_date  |  location
-----------+------------+-------------
       450 | 2019-03-04 | kannur
       452 | 2016-06-07 | kozhikode
       454 | 2015-01-03 | kochi
       453 | 2014-02-03 | trivamndrum
       457 | 2017-10-09 | kollam
(5 rows)

albin=# insert into owns values('KL12','KA24');
INSERT 0 1
albin=# insert into owns values('KL13','KA21');
INSERT 0 1
albin=# insert into owns values('KL14','KA54');
INSERT 0 1
```

```
albin=# insert into owns values('KL15','KA34');
INSERT 0 1
albin=# insert into owns values('KL16','KA87');
INSERT 0 1
albin=# select * from person;
 driver_id |  name   |  address
-----------+---------+-----------
 KL13      | arun    | odanthodu
 KL12      | rajan   | manathana
 KL14      | nafseen | peravoor
 KL15      | nidhin  | irity
 KL16      | ashin   | kannur
(5 rows)

albin=# insert into participated values('KL12','KA24',450,34000);
INSERT 0 1
albin=# insert into participated values('KL13','KA21',452,6790);
INSERT 0 1
albin=# insert into participated values('KL14','KA54',454,7800);
INSERT 0 1
albin=# insert into participated values('KL15','KA34',453,56900);
INSERT 0 1
albin=# insert into participated values('KL16','KA87',457,67844);
INSERT 0 1
```

albin=# select * from participated;

driver_id | regno | report_number | damage_amout

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

KL12 | KA24 | 450 | 34000

KL13 | KA21 | 452 | 6790

KL14 | KA54 | 454 | 7800

KL15 | KA34 | 453 | 56900

KL16 | KA87 | 457 | 67844

(5 rows)

3. Update the damage amount for the carwith specific regno in the accident with report number 454 to 25000

albin=# update participated set damage_amout=25000 where report_number=454 and regno='KA54';

UPDATE 1

4. Show the contents of participated table.

albin=# select * from participated;

driver_id | regno | report_number | damage_amout

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

KL12 | KA24 | 450 | 34000

KL13 | KA21 | 452 | 6790

KL15 | KA34 | 453 | 56900

KL16 | KA87 | 457 | 67844

KL14 | KA54 | 454 | 25000

(5 rows)

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

```
albin=# select count(*) as totcar from car,participated where
car.regno=participated.regno and car.model='BENZ';
 totcar
--------
      1
(1 row)
```

6. Find the total number of people who owned cars that were involved in accidents in the year 2018.

```
albin=# select count(driver_id) as people from owns e,car c where c.year=2018 and
e.regno=c.regno;
 people
--------
      1
(1 row)
```

17. BANKING ENTERPRISE

Consider the following database for a banking enterprise.

BRANCH(**branch_name**:string , branch_city:string , assets:real)

ACCOUNT(**accno**:int , branch_name:string , balance:real)

DEPOSITOR1(**customer_name**:string , **accno**:int)

CUSTOMER1(**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.**

albin=# 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

albin=# 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

albin=# create table customer1(customer_name varchar(20) primary key,customer_street varchar(20),customer_city varchar(20));

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "customer1_pkey" for table "customer1"

CREATE TABLE

albin=# create table depositor1(acc_no int references account,customer_name varchar(20) references customer1);

CREATE TABLE

albin=# 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

albin=# create table borrower(customer_name varchar(20) references customer1,loan_no int references loan);

CREATE TABLE

2. Enter at least five tuples for each relation

albin=# insert into branch values('kannur','kerala',50000);

INSERT 0 1

albin=# insert into branch values('thamilnadu','chennai',40000);

INSERT 0 1

albin=# insert into branch values('kochi','kerala',30000);

INSERT 0 1

albin=# insert into branch values('calicut','kerala',35000);

INSERT 0 1

albin=# insert into branch values('kasargod','kerala',25000);

INSERT 0 1

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

INSERT 0 1

albin=# insert into account values(102,'thamilnadu',25000);

```
INSERT 0 1

albin=# insert into account values(103,'kochi',15000);

INSERT 0 1

albin=# insert into account values(104,'calicut',20000);

INSERT 0 1

albin=# insert into account values(105,'kasargod',25000);

INSERT 0 1

albin=#  insert into customer1 values('albin','kannur','kerala');

INSERT 0 1

albin=#  insert into customer1 values('ashin','thamilnadu','chennai');

INSERT 0 1

albin=#  insert into customer1 values('swetha','kochi','kerala');

INSERT 0 1

albin=#  insert into customer1 values('rini','calicut','kerala');

INSERT 0 1

albin=#  insert into customer1 values('neena','kasargod','kerala');

INSERT 0 1

albin=# insert into depositor1 values(101,'albin');

INSERT 0 1

albin=# insert into depositor1 values(102,'ashin');

INSERT 0 1

albin=# insert into depositor1 values(103,'swetha');

INSERT 0 1

albin=# insert into depositor1 values(104,'rini');
```

```
INSERT 0 1

albin=# insert into depositor1 values(105,'neena');

INSERT 0 1

albin=# insert into loan values(1001,'kannur',10000.00);

INSERT 0 1

albin=# insert into loan values(1002,'thamilnadu',20000.00);

INSERT 0 1

albin=# insert into loan values(1003,'kochi',25000.00);

INSERT 0 1

albin=# insert into loan values(1004,'calicut',35000.00);

INSERT 0 1

albin=# insert into loan values(1005,'kasargod',30000.00);

INSERT 0 1

albin=# insert into borrower values('albin',1001);

INSERT 0 1

albin=# insert into borrower values('ashin',1002);

INSERT 0 1

albin=# insert into borrower values('swetha',1003);

INSERT 0 1

albin=# insert into borrower values('rini',1004);

INSERT 0 1

albin=# insert into borrower values('neena',1005);

INSERT 0 1
```

3.Show the contents of each table.

```
albin=# select * from branch;
```

```
 branch_name | branch_city |  assets
-------------+------------+---------
 kannur      | kerala      | 50000.00
 thamilnadu  | chennai     | 40000.00
 kochi       | kerala      | 30000.00
 calicut     | kerala      | 35000.00
 kasargod    | kerala      | 25000.00
(5 rows)
albin=# select * from account;
 acc_no | branch_name | balance
--------+------------+---------
    101 | kannur      | 15000.00
    102 | thamilnadu  | 25000.00
    103 | kochi       | 15000.00
    104 | calicut     | 20000.00
    105 | kasargod    | 25000.00
(5 rows)
albin=# select * from customer1;
 customer_name | customer_street | customer_city
---------------+----------------+--------------
 albin         | kannur          | kerala
 ashin         | thamilnadu      | chennai
 swetha        | kochi           | kerala
 rini          | calicut         | kerala
```

```
 neena        | kasargod        | kerala
(5 rows
albin=# select * from depositor1;
 acc_no | customer_name
--------+--------------
    101 | albin
    102 | ashin
    103 | swetha
    104 | rini
    105 | neena
(5 rows)
albin=# select * from loan;
 loan_no | branch_name | account
---------+------------+---------
    1001 | kannur      | 10000.00
    1002 | thamilnadu  | 20000.00
    1003 | kochi       | 25000.00
    1004 | calicut     | 35000.00
    1005 | kasargod    | 30000.00
(5 rows)
albin=# select * from borrower;
 customer_name | loan_no
---------------+--------
 albin         |    1001
```

ashin | 1002

swetha | 1003

rini | 1004

neena | 1005

(5 rows)

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

albin=# select d.customer_name from depositor1 d,account a where a.acc_no=d.acc_no and a.branch_name='kochi' group by d.customer_name having count(*)>=1;

customer_name

---

swetha

(1 row)

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

albin=# 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='kochi');

DELETE 1

albin=# select * from account;

acc_no | branch_name | balance

---+---+---

101 | kannur | 15000.00

102 | thamilnadu | 25000.00

104 | calicut | 20000.00

105 | kasargod | 25000.00

(4 rows)

18. 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.

albin=# 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

albin=# 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

albin=# 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

albin=# 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

```
albin=# 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.

```
albin=# insert into students values('dbc101','Albin','major','10/11/1997');
INSERT 0 1
albin=# insert into students values('dbc102','Shyam','major','02/05/1996');
INSERT 0 1
albin=# insert into students values('dbc103','Vyshnav','major','05/15/1998');
INSERT 0 1
albin=# insert into students values('dbc104','Amrutha','major','11/02/1997');
INSERT 0 1
albin=# insert into students values('dbc105','Jalna','major','01/06/1995');
INSERT 0 1

albin=# insert into course values(201,'BCA','CS');
INSERT 0 1
albin=# insert into course values(202,'Bsc','MATHS');
INSERT 0 1
albin=# insert into course values(203,'Bsc','PHYSICS');
INSERT 0 1
albin=# insert into course values(204,'MCA','CS');
INSERT 0 1
albin=# insert into course values(205,'BCA','CS');
INSERT 0 1
albin=# insert into text values(111,'C','DC','Balaguru');
INSERT 0 1
albin=# insert into text values(112,'Management','DD','Gupta');
INSERT 0 1
```

```
albin=# insert into text values(113,'finance','DD','dev');
INSERT 0 1
albin=# insert into text values(114,'cc','DC','kapil');
INSERT 0 1
albin=# insert into text values(115,'CG','DD','albert');
INSERT 0 1


albin=# insert into enroll values('dbc101',201,1,50);
INSERT 0 1
albin=# insert into enroll values('dbc102',202,2,40);
INSERT 0 1
albin=# insert into enroll values('dbc103',203,1,35);
INSERT 0 1
albin=# insert into enroll values('dbc104',204,2,40);
INSERT 0 1
albin=# insert into enroll values('dbc105',205,1,30);
INSERT 0 1
albin=# insert into book_adoption values(201,1,111);
INSERT 0 1
albin=# insert into book_adoption values(202,2,112);
INSERT 0 1
albin=# insert into book_adoption values(203,1,113);
INSERT 0 1
albin=# insert into book_adoption values(204,2,114);
INSERT 0 1
albin=# insert into book_adoption values(205,1,115);
INSERT 0 1
```

3. Show the contents of each table.

```
albin=# select * from students;
 regno  | bame  | major |   bdate
--------+------+------+-----------
```

```
 dbc101 | Albin  | major | 1997-10-11
 dbc102 | Shyam | major | 1996-02-05
 dbc103 | Vyshnav  | major | 1998-05-15
 dbc104 | Amrutha   | major | 1997-11-02
 dbc105 | Jalna  | major | 1995-01-06
(5 rows)


albin=# select * from course;
 courseno | cname |  dept
----------+------+--------
      201 | BCA   | CS
      202 | Bsc   | MATHS
      203 | Bsc   | PHYSICS
      204 | MCA   | CS
      205 | BCA   | CS
(5 rows)


albin=# select * from text;
 book_isbn | book_title | publisher |  author
-----------+-----------+----------+---------
       111 | C         | DC       | Balaguru
       112 | Management | DD       | Gupta
       113 | finance    | DD       | dev
       114 | cc         | DC       | kapil
       115 | CG         | DD       | albert
(5 rows)


albin=# SELECT * FROM ENROLL;
 regno  | courseno | sem | marks
--------+---------+----+------
 dbc101 |      201 |   1 |    50
 dbc102 |      202 |   2 |    40
```

```
 dbc103 |     203 |   1 |    35
 dbc104 |     204 |   2 |    40
 dbc105 |     205 |   1 |    30
(5 rows)
```

```
albin=# select * from book_adoption;
 courseno | sem | book_isbn
----------+----+----------
     201 |   1 |      111
     202 |   2 |      112
     203 |   1 |      113
     204 |   2 |      114
     205 |   1 |      115
(5 rows)
```

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

```
albin=# 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
---------
 MATHS
 PHYSICS
 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.

```
albin=# 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
----------+----------+-----------
      201 |      111 | C
      204 |      114 | cc
      205 |      115 | CG
(3 rows)
```

LINUX OPERATING SYSTEM

USER MANAGEMANT COMMANDS

COMMAND

useradd - Create a new user or update default new user information.

SYNTAX

useradd [-c comment] [-d home_dir]

[-e expire_date] [-f inactive_time]

[-g initial_group] [-G group[,...]]

[-m [-k skeleton_dir] | -M] [-n] [-o] [-p passwd] [-r]

[-s shell] [-u uid] login

DESCRIPTION

Creating New Users

When invoked without the -D option, the useradd command creates a new user account using the values specified on the command line and the default values from the system.

The options which apply to the useradd command are:

-c comment

The new user's password file comment field.

-d home_dir

The new user will be created using home_dir as the value for the user's login directory. The default is to append the login name to default_ home and use that as the login directory name.

COMMAND

login - sign on

SYNTAX

login [name]

login -p

login -h hostname

login -f name

DESCRIPTION

login is used when signing onto a system. It can also be used to switch from one user to another at any time. If an argument is not given, login prompts for the username.

COMMAND

passwd - update a user's authentication token(s).

SYNTAX

passwd [-k] [-l] [-u [-f]] [-d] [-n mindays] [-x maxdays] [-w warndays] [-i inactivedays] [-S] [username]

OPTIONS

-k The option, -k, is used to indicate that the update should only be for expired authentication tokens(passwords); the user wishes to keep their non-expired tokens as before.

-l This option is used to lock the specified account and it is available to root only.

COMMAND

hostname - show or set the system\x{2019}s host name

SYNTAX

hostname [-v] [-a] [--alias] [-d] [--domain] [-f] [--fqdn] [-i] [--ip-address] [--long] [-s] [--short] [-y] [--yp] [--nis] [-n] [--node]

hostname [-v] [-F filename] [--file filename] [hostname]

DESCRIPTION

Hostname is the program that is used to either set or display the current host, domain or node name of the system. These names are used by many of the networking programs to identify the machine. The domain name is also used by NIS/YP.

GET NAME

When called without any arguments, the programs displays he current names:hostname will print the name of the system as returned by the gethostname(2) function.

COMMAND

logout - sign out

logout [status] - Exit the shell, returning status to invoking program if

Specified. Can be used only in a login shell. Otherwise, use exit.

COMMAND

shell - creates and manipulate a shell widget.

SYNTAX

shell pathName ? options?

INHERITANCE

itk::Toplevel <- shell

STANDARD OPTIONS

background cursor foreground

DESCRIPTION

The shell command creates a shell which is a top level which supports modal operation.

METHODS

The shell command creates a new Tcl command whose name is pathname. This command may be used to invoke various operations on the widget.

DIRECTORY AND FILE RELATED COMMANDS

COMMAND

mkdir - make directories

SYNTAX

mkdir [OPTION] DIRECTORY...

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

-m, --mode=MODE

set permission mode (as in chmod), not rwxrwxrwx - umask

-p, --parents

no error if existing, make parent directories as needed.

COMMAND

cd [-L|-P][dir]

Change the current directory to directory. The variable HOME is the default dir. The variable CDPATH defines the search path for the directory containing dir, Alternative directory names in CDPATH are separated by a colon (:)

COMMAND

cat - concatenate files and print on the standard output

SYNTAX

cat [OPTION] [FILE]...

DESCRIPTION

Concatenate FILE(s), or standard input, to standard output.

-A, --show-all

equivalent to -vET

-b, --number-nonblank

number nonblank output lines

COMMAND

pwd - print name of current/working directory.

SYNTAX

pwd [option]

Print the full file name of the current working directory.

COMMAND

file - determine the file type

SYNTAX

file [-bciknsvzL][-f namefile][-m magicfiles] file

file -C [-m magicfile]

DESCRIPTION

File tests each argument in an attempt to classify it. There are three sets of tests, perform in this order: file system tests, magic number tests and language tests. The first test that succeeds causes the file type to be printed.

OPTIONS

-b do not prepend file names to output lines(brief mode).

-c cause a checking printout of the parsed form of the magic file. This

is usually used in conjunction with –m to debug a new magic file

before installing it.

-C write a magic.mgc output file that contains a pre parsed version of

file.

COMMAND

tree - list contents of directories in a tree-like format.

SYNTAX

tree [-adfghilnpqstuxACDFN][-P pattern][-I pattern][directory]

DESCRIPTION

Tree is a recursive directory listing program that produces a depth intended listing of files. Upon completion of listing all files/directories found, tree returns the total number of files and/or directories listed.

COMMAND

cp - copy files and directories

SYNTAX

cp [OPTION]... SOURCE DEST

cp [OPTION]... SOURCE... DIRECTORY

cp [OPTION]... --target-directory=DIRECTORY SOURCE...

DESCRIPTION

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY. Mandatory arguments to long options are mandatory for short options too.

-a, --archive

same as -dpR

--backup[=CONTROL]

make a backup of each existing destination file

-b like --backup but does not accept an argument

COMMAND

mv - move (rename) files

SYNTAX

mv [OPTION]... SOURCE DEST

mv [OPTION]... SOURCE... DIRECTORY

mv [OPTION]... --target-directory=DIRECTORY SOURCE...

DESCRIPTION

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

COMMAND

more - file perusal filter for crt viewing

SYNTAX

more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]

DESCRIPTION

More is a filter for paging through text one screen full at a time. This

version is especially primitive. Users should realize that less (1) provides more (1) emulation and extensive enhancements.

OPTIONS

-num This option specifies an integer which is the screen size (in lines).

-d more will prompt the user with the message "[Press space to continue, 'q' to quit.]" and will display "[Press 'h' for instructions.]"instead of ringing the bell when an illegal key is pressed.

-l more usually treats ^L (form feed) as a special character, and will pause after any line that contains a form feed. The -l option will prevent this behavior.

COMMAND

rm - remove files or directories

SYNTAX

rm [OPTION]... FILE...

DESCRIPTION

This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories.

If a file is unwritable, the standard input is a tty, and the -f or --force option is not given, rm prompts the user for whether to remove the file. If the response does not begin with ‘y’ or ‘Y’, the file is skipped.

OPTIONS

Remove (unlink) the FILE(s).

-d, --directory

unlink FILE, even if it is a non-empty directory

(super-user only)

-f, --force

Ignore nonexistent files, never prompt

-i, --interactive

Prompt before any removal.

COMMAND

man - format and display the on-line manual pages

manpath - determine user’s search path for man pages

SYNTAX

man [-acdfFhkKtwW] [--path] [-m system] [-p string] [-C config_file]

[-M pathlist] [-P pager] [-S section_list] [section] name ...

DESCRIPTION

man formats and displays the on-line manual pages. If you specify section, man only looks in that section of the manual. name is normally the name of the manual page, which is typically the name of a command, function, or file.

However, if name contains a slash (/) then man interprets it as a file specification, so that you can do man. /foo.5 or even man /cd/foo/bar.1.gz.

OPTIONS

-C config_file

Specify the configuration file to use; the default is

/etc/man.config.

-M path

Specify the list of directories to search for man pages. Sepa-

rate the directories with colons. An empty list is the same as not specifying -M at all. See SEARCH PATH FOR MANUAL PAGES.

WILDCARDS

File names are the most common arguments used in a command. Often, you may know only part of the file name or you may want to references several file names that have the same extension or begin with the same characters. The shell provides a set of special characters called wildcards that search out, match and generate a list of file names. Wild characters are asterisk, the question mark and brackets (*,?, []) .

WILDCARD SYMBOLS

* Match on any set of characters in file names

? Match on any single character in file name

[ ] Match on class

SEARCH PATH FOR MANUAL PAGES

man uses a sophisticated method of finding manual page files, based on the invocation options and environment variables, the /etc/man.config configuration file, and some built in conventions and heuristics.

First of all, when the name argument to man contains a slash (/), man assumes it is a file specification itself, and there is no searching involved.

PATHNAME:

Specifies the path of the current working directory. Suppose that ‘data’ is the current working directory, then we have specifies it ‘/home/data’.

RELATIVE PATHNAME:

Specify only the full path name from the current directory

Eg- To refer directory’ data’, we just specify only’/data’.

ABSALUTE PATH NAME

Specify the full path name. Suppose, ‘data’ contained in dir. ‘Linux’, then to specify ‘data’, we use the full path as follows,

‘/home/Linux/data’.

REFERRING TO HOME DIRECTORIES

1: $:HOME: Environment variable store our home directory name.

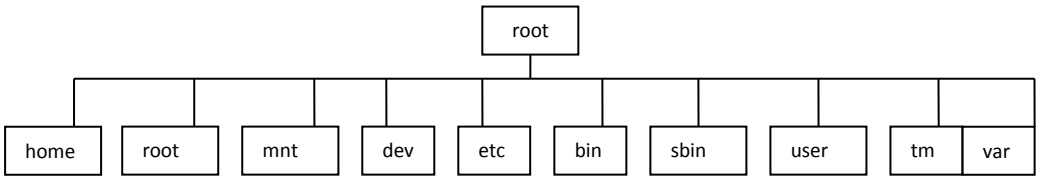
2: ~: The tilde represents our home directory name.

3:’.’ : Single dot represent current working directory.

4:’**..** ’: Double dot refer to a directly above the current directory.

5:OLD PWD: It refers to the previous working directory before you change to the new one.

FILES AND DIRECTORIES



/bin: Contain Common LINUX user commands such as ls, date and chmod.

/dev: Contains files representing access points to device on your systems.

/etc: Contains administrative configuration files.

/home: Contains directories assigned to each user with a log in account.

/mnt: Provides allocation for mounting devices such as remote file system and removable media.

/root: Represents root user's home directory.

/sbin: Represents administrative files and the daemon process.

/tmp: Contains temporary files used by the applications.

/usr: Contains user documentation, games, graphical files(x11), libraries (libo) and a variety of other user and administrative commands and files.

/boot: Have the bootable Linux kernel and boot loader configuration files.

/var: Contains directories of date used by various applications. In particular, this is where you would place file that you share as an FTP server or a web server.

CHANGING PERMISSIONS

NAME

chmod – change file access permissions

SYNTAX

chmod [OPTION]…MODE[,MODE]…FILE…

chmod [OPTION]…OCTAL-MODE FILE…

chmod[OPTION] …--reference=RFILE FILE…

DESCRIPTION

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions.chmod never changes the

permissions of symbolic links; the chmod system call can not change their permissions. This is not a problem since the permissions of symbolic links are never used. However, each symbolic link listed on the command line, chmod changes the permissions of the pointed-to file. In contrast, chmod ignores symbolic links encountered during recursive directory traversals.

OPTION

Change the mode of each FILE to MODE.

-c, --changes

like verbose but report only when a change is made

-f, --silent, --quiet

like suppress most error messages

-v, --verbose

output a diagnostic for every file processed

-R, --recursive

change files and directories recursively

--help

display this help and exit

--version

output version information and exit

Vi EDITOR

Although many different kinds of editors may be available on any given Linux system, all systems have the two standard editors: Ed and Vi .Vi, which stands for "visual", remains one of the most widely used editors in Linux. The vi editor displays a wall screen of data at a time and allows you to edit any data shown on the screen.

Editors use a key board for two very different operations: to specify editing commands and to receive character input. As editing commands, certain keys will perform deletions, some will execute changes, and others will perform cursor movement. Instead of dividing the command and input functions among different keys, the Vi editor has two separate modes of operations for the key board: command mode and input mode. In command mode, all the keys on the key board become editing commands. In the input mode, the keys on th4e key board become input characters.

When you change modes, the functionality of the key board changes. When you invoke the Vi editor, you are placed in command mode. Each key know becomes an editing command.

Once in the input mode, the key board again changes functionality. Each key now represents a character to be input to the text. The keyboard becomes like a type writer.

Though the Vi command mode handles most editing operations, there are some, such as file saving and global substitutions, that it cannot perform. for such operations you need to execute line editing commands. you enter the line editing mode using the Vi colon command,:. The colon is a special command that allows you to perform one line editing operation. Upon pressing the colon, a line opens up at the bottom of the screen with the cursor placed at the beginning of the line.you are know in the line editing mode. In this mode,you enter an editing command on a line, press ENTER, and the command is executed. Entry into this mode is only temporary. Upon pressing ENTER, you are automatically reurned to the Vi command mode, and the cursor returns to its previous position on this screen.

The command and input operations constitute two very separate modes.Add to this line editing mode,and you are faced to with three very different modes of operations in Vi.The line editing mode operates on the line : you type in a command with its arguments and terminate the command the command by pressing ENTER.The Vi command mode,however ,operates by single keys.Simply pressing a key or sequence of keys executes an editor command.The Vi input mode input characters into a text file.Any key is a valid character for except for ESC.The ESC returns you to the command mode.The Vi editor you can create,save,close and quit

files.The commands for each are not all that similar. Saving and Quiting a file involves the use of special line editing commands, where as closing a file is a Vi editing command. Creation of a file is usually specified on the same shell command line that invokes the Vi editor.

To editor file,type Vi and the name of the file on the shell command line.If a file by that name does not exist the system will create it.In effect ,giving the name of the file that does not exist instructs the Vi editor to create that file.

On Linux vi command is usually a link to one of these programs.

Vi filename

Once the file is opened, you are in command mode. After all there are three modes together.

- Command mode
- Insert mode
- Exmode

The vi program has 3 modes of operation:

Command Mode: Here all the keys pressed by the user are interpreted to be editor commands

Insert Mode: It permits us to give commands at the command line. The bottom line of the vi screen is called the command line.

Key	Cursor Movement
h	Moves cursor left one character
l	Moves cursor right one character
k	Moves cursor up one line
j	Moves cursor down one line
w	Moves cursor forward one line
W	Moves cursor forward one space delimited word
b	Moves cursor backward one word

B Moves cursor back one space delimited word

e Moves cursor to the end of the next word

E Moves cursor to the end of the next space delimited word

o Moves cursor to the beginning of the line

$ Moves cursor to the end of the line

ENTER Moves cursor to beginning of the next line

~ Moves cursor to beginning of previous line

(Moves cursor to beginning of sentence

) Moves cursor to end of sentence; successive command moves to beginning of next sentence

{ Moves cursor to beginning of paragraph

} Moves cursor to end of paragraph

CTRL-F Moves forward by a screen of text; the next screen of text is displayed

CTRL-B Moves backward by a screen of text; the previous screen is displayed

CTRL-D Moves forward by one-half screen of text

CTRL-U Moves backward by one-half screen of text

G Moves cursor to last line in the text

numG Moves cursor to specific line number 45G will place the cursor on line 45

H Moves cursor to line displayed on screen

M Moves cursor to middle line displayed on screen

,, Moves the cursor to its previous location in the text

mmark Places a mark on a line of text; the mark can be any alphabetic character

'mark Moves the cursor to the line with the mark

Input All input commands place the user in input; the user leaves input with ESC

a Enters input after the cursor

A Enters input at the end of a line

i Enters input before the cursor

I Enters input at the beginning of a line

o Enters input below the line the cursor is on; insert a new empty line below the one the cursor is currently on

O Enters input above the line the cursor is on; insert a new empty line above the one the cursor is currently on

Delete

x Deletes the character the cursor is on

X Deletes the character before the character the cursor is on

dw Deletes the word the cursor is on

db Deletes the beginning of a word

dW Deletes space delimited word

dB Deletes to beginning of a space delimited word

dd Deletes the line the cursor is on

D Deletes the rest of the line the cursor is on

d0 Deletes text from cursor to beginning of line

d Deletes following text specified

d) Deletes rest of a sentence

d} Deletes rest of a paragraph

dG Deletes rest of the file

dm Followed by a mark, deletes everything to mark

dL Deletes the rest of screen

dH Deletes top of the screen

J Joins the line below the cursor to the end of the current line; in effect, deleting the new line character of the line the cursor is on

Change Except for the replace command, x, all commands place the user into input after deleting text

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

r Replace the character the cursor is on; after pressing **x** the user enters the replacement character; the change is made without entering input; the user remains in the Vi command mode

R First places into the input mode, then overwrites character by character; appears as an overwrite mode on the screen but actually is in input mode

Move Moves text by first deleting it, moving the cursor to desired place of insertion, and then pressing the **p** command. (When text is deleted, it is automatically held in a special buffer.)

P Inserts deleted or copied text after the character or line the cursor is on

P Inserts deleted or copied text before the character or line the cursor is on

dw p Deletes a word, then moves it to the place you indicate with the cursor (press **p** to insert the word *after* the word the cursor is on)

dw p Deletes a word, then moves it to the place you indicate with the cursor (press p to insert the word *before* the word the cursor is on

dd p Deletes a line, then moves it to the place you indicate with the cursor (press **p** to insert the word *after* the line the cursor is on)

d p Deletes following text specified, then moves it to the place you indicate with the cursor(press **p** or **P**)

d) p Moves the rest of a sentence

d} p Moves the rest of a paragraph

dG p Moves the rest of the file

dm p Followed by a mark, moves everything to mark

dL p Moves the rest of the screen

File Operation		**Effect**
W	Write	Saves file
r	filename Read	Inserts file text
q	Quit	Quits editor

Delete, Move,And Copy

d	Delete	Deletes a line or a set of lines m*Num*

Move	Move a line or a set of lines by deleting them and then inserting them after line *Num* co*Num*
Copy	Copies a line or a set of lines by copying them and then inserting the copied text after line *Num*

Line Reference		**Description**
Num	Line number	A number references that line number
Num, Num	Set of lines	Two numbers separated by a Comma references a set of lines

Special Characters

Any character		Matches on any one possible character in a pattern
*Repeated	chars	Matches 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

yy or Y Cop[ies the line the corsor is on,then moves the line to the place you indicate with the cursor (press p to insert after the line the cursor is on)

y Copies following text specified

y) Copies the rest of a sentence

y} Copies the rest of a paragraph

yG Copies therest of the file

ym Followed by a mark ,copies everything to mark.

yL Copies the rest of the screen

yH Copies to the top of the screen

THE LINUX SHELLS

The shell is a program that acts as a buffer between you and the operating system. In its role as a command interpreter, it should act invisibly. It also can be used for simple programming.

PURPOSE OF A SHELL

There are 3 main uses for the shell,

1) Interactive use.
2) Customization of your Linux session.
3) Programming.

Program name	Shell
/bin/sh /bin/bash /bin/csh /bin/tcsh	Bourne again shell Bourne again shell C shell or [tcsh] tcsh

Shell flavors

Many different Linux shells are available:

- the Bourne-Again shell [bash] that is based on the Bourne shell [sh] and is standard for Linux.
- the C shell [csh] that uses c syntax and has many conveniences.
- tcsh, an extension of the csh that appears instead of csh in many Linux distributions.

Bash: The Bourne again shell

Bash is the GNU version of the standard Bourne shell- the original Unix shell – and incorporates many popular features from the other shells such as sh, tcshand the Korn shell [ksh]. If execute as part of the user's login, bash starts by executing any commands found in ~/.bash_profile, or ~/.bash_login.

csh and tcsh

on some versions of Linux, tcsh is used as C shell.

MOUNTING FILE SYSTEMS

COMMAND

mount - mount the file system.

SYNTAX

mount [-lhV]

mount -a [-fFnrsvw] [-t vfstype] [-O optlist]

mount [-fnrsvw] [-o options[,….]] device | dir

mount [-fnrsvw] [-t vfstype] [-O options] device dir

DESCRIPTION

All files accessible in a UNIX system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree. Conversely, the 'umount' command will detach it again.The standard form of mount command, is

mount -t type device dir

OPTIONS

-V output version.

-h print a help message.

-v verbose mode.

-a mount all file systems (of the given types) mentioned in fstab.

-F (used in conjunction with –a). Fork off a new incarnation of mount for each device of possible characters in a file name.

COMMAND

umout-Unmount the file system

SYNTAX

Umount [-dflnrv][-t vfstype][-o option]

DESCRIPTION

The umount command detaches the file system(s) mentioned from the file heirachy. A file system is specified by giving the directory where it has been mounted.

OPTION

-n Umount without writing in /etc/mtab

-r In case unmounting fails, try to remount read-only.

FILTER

Filters are commands that read data, perform operations on that data, and then send result to the standard output. Filters generate different kinds of output, depending on their task. Filter can be operating on a stream of data. As data is passed through the filter, it is analyzed, screened or modified. The data output stream to a filter consists of a sequence of bytes that can be received from the files, devices or a output of other commands or filters. The filter operates on a stream of a data, but it does not operate (or modify) the source of data. Only files data is read and fed into filter. The output of a filter is usually sent to the standard output. It can be then directed to the file or device, or piped as input to another utility or filter.

A filter is any command that gets its input from standard input, manipulates the lines, and then send them to standard output. Three filters – grep, sed, awk, - are so powerful. There are some common filters: cat, cmp, comm., cut, diff, head, paste, sort, tail, tr, unique and wc.

COMMON FILTERS

FilterAction

more passes all data from input to output, with pauses at the end of each screen of 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

hea passes the number of specified lines at the beginning of data

paste c ombines columns

sort arranges the data in sequence

tail passes the number of specified lines at the end of the data

tr translate one or more characters as specified

uniq deletes duplicate (repeated) lines

wc counts characters, words or lines

sed passes edited lines

awk passes edited lines – passes lines

INPUT OUTPUT REDIRECTION

Standard input /output and redirection

When UNIX was designed, a decision was made to distinguish between the physical implementation and logical organization of a file. This logical file organization extends to input and output operations. The data in input and output is organized like file. Data input at the keyboard is placed in a data stream arranged as a set of bytes. Data output from a command or program is also placed in a data stream and arranged as a continuous set of bytes. This input data stream is referred to in Linux as the standard input and output data stream is called the standard output. Because the standard input and standards output have the same organization as that of a file. Linux as redirection capability that lets you easily move data in out of files.

Redirecting the standard output :> and >>

If you want to direct the standard output to a file rather than the screen to this you place the output redirection operator ,> and name of a file on the command line after the linux command

$ cat myletter > newletter

The redirection operation creates the new destination file. If the file already exists it will be over written with the data in the standard output.

Appending the standard output :>>

We can also append the standard output to an existing file using the >> redirection operator.Instead of overwriting the file , the data in the standard output is added at the end of the file.

$ cat myletter >> alletters

$ cat oldletter >> alletters

Redirecting the standard input:<

The standard input may be received from the file rather than the keyboard. The operator for redirecting the standard input is less than sign, <.

$ cat < myletter

don Bosco college, angadikadavu

$

Using redirection and pipes with filters

Filters send their output to the standard output and so, by default, they display their output on the screen. The simplest filters merely output contents of files. We can save the output of a filter in a file using redirection or pipes. Save for e. g.: to print we use

$ cat complist | Ipr

PIPING

The UNIX piping facility let us connect commands to other commands. This facility is of at most importance in combining Unix commands and operations if can be really useful to redirect the output of one program so that it become the input of another program, there by joining two programs to send the output of one command as input of another the two command must be joined using a pipe (|) character.The pipe operator,|, placed between two command form a connection between them.

$ ls *.c | lpr :-To generate the list of file names with .c extension, then this list is piped to the lpr (to the printer) command.

$ cat | lpr

This text will be printed ^D$:-This example cat first read the input from the keyboard instead of file and pipes the output to the lpr command.

Pipes and Redirection : tee

If you want to redirect the standard output to a file and, at the same time, display the content of the output on the screen so that you can see what you are saving. You can do this with tee command.

$ sort mylist | tee sfile

computer

modem

screen

$

SHELL PROGRAMS

1.Find the factorial of a given number

Aim:

Create a program to find the factorial of a given number

Algorithm:

Step1 : start

Step2 : read number n

Step3 : set f=1 and j=1

Step4 : if j less than n go to step 5 otherwise go to step 6

Step5 : fact=fact* j , j=j+1 go to step 4

Step6 : print value of f

Step7 : stop

Program:

```
echo "Enter the number"
read n
f=1
j=1
while [ $j -le $n ]
do
    f=`expr $f \* $j`
    j=`expr $j + 1`
done
echo "Factorial = $f"
```

Output:

Enter the number 5

Factorial = 120

.

2.Find Fibonacci Series

Aim:

Create a program to print Fibonacci series up to a given number entered as command line

Algorithm:

Step 1 : start

Step 2 : read n

Step 3 : if n eqyal to 0 go to step 4 otherwise go to step 5

Step 4 : print value 0

Step 5 : set a = 0 and b =1 k =1

Step 6 : print a and b

Step 7 : set i = n-2

Step 8 : if k less than i go to step 9 otherwise go to step 13

Step 9 : set c = a+b

Step 10 : print c

Step 11 : set a = b,b = c

Step 12 : k = k+1 go to step 8

Step 13 : stop

Program:

```
echo "Enter the limit"
read n
if [ $n -eq 0 ]
then
    echo 0
    exit
fi
```

```
a=0
b=1
k=1
echo "The fibnocci series"
echo $a
echo $b
i=`expr $n-2`
while [[ $k -le $i ]]
do
    c=`expr $a + $b`
    echo $c
a=$b
b=$c
k=`expr $k + 1`
done
```

Output:

```
Enter the limit
6
The fibnocci series
0
1
1
2
3
5
```

3.Find the reverse of a number.

Aim:

Create a program to print the reverse of a number.

Algorithm:

Step 1 : start

Step 2 : read number num

Step 3 : set num2=0

Step 4 : if num not equal to 0 go to step 5 other wise go to step 7

Step 5 : calculate num2=num % 10 + num2 * 10

Step 6 : calculate num=num/10 go to step 4

Step 7 : set print num2

Step 8 : stop

Program:

```
echo “Enter number”
read num
num2=0
echo “Reverse is”
while [ $num -ne 0 ]
do
num2=` expr $num % 10 + $num2 \* 10 `
num=` expr $num / 10 `
done
echo “$num2”
```

Output:

Enter number

234

Reverse is

432

4.To check largest among three numbers.

Aim :

Create a program to find the largest among three numbers.

Algorithm:

Step 1 : start

Step 2 : read numbers a,b,c

Step 3 : set l=a

Step 4 : if b greater than l go to step 5 otherwise go to step 6

Step 5 : set l=b go to step 8

Step 6 : if c greater than l go to step 7 otherwise go to step 6

Step 7 : set l=c

Step 8 : print value of l

Step 9 : stop

Program:

```
echo “Enter three numbers”

read a b c

l=$a

if [ $b -gt $l ]

then

l=$b

fi

if [ $c -gt $l ]

then
```

```
l=$c

fi

echo “Largest among $a $b and $c is $l”
```

Output:

```
Enter three numbers

23 56 8

Largest among 23 56 and 8 is 56
```

5.Find even and odd numbers

Aim:

Create a program to find evan and odd numbers.

Algorithm:

Step 1 : start

Step 2 : read number n

Step 3 : set b=n%2

Step 4 : if b equal to 0 go to step 5 otherwise go to step 6

Step 5 : print the value of n is even

Step 6 : print the value of n is odd

Step 7 : stop

Program:

```
echo “Enter the number”

read n

b=$[ n%2]

if [ $b -eq 0 ]

then

echo “Given number $n is even”

else

echo “Given number $n is odd”

fi
```

Output:

Enter the number

34

Given number 34 is even

6. Swapping two numbers

Aim:

Create a program to swap two numbers.

Algorithm:

Step 1 : start

Step 2 : read a,b

Step 3 : print the value of a and b

Step 4 : set t=0, t=a, a=b, b=t

Step 5 : print the value of a and b

Step 6 : stop

Program:

```
echo “Enter the values”

read a b

echo “Before swap value of a and b $a $b”

t=0

t=$a

a=$b

b=$t

echo “After swap value of a and b $a $b”
```

Output:

Enter the values

34 45

Before swap value of a and b 34 45

After swap value of a and b 45 34