```
function noofrows(m:matrix):integer;
begin
     noofrows:=trunc(m[0,1])
end;

function noofcols(m:matrix):integer;
begin
     noofcols:=trunc(m[1,0])
end;

procedure entermatrix(var m:matrix);
var r,c:integer;
begin
     write('Enter no. of rows   : ');
     readln(input,m[0,1]);
     write('Enter no. of columns: ');
     readln(input,m[1,0]);
     for r:=2 to noofcols(m) do
         m[0,r]:=m[0,1];
     for c:=2 to noofrows(m) do
         m[c,0]:=m[1,0];
     for r:=1 to noofrows(m) do
     begin
         write('Enter row ',r:1,' :');
         for c:= 1 to noofcols(m) do
         begin
              read(input,m[r,c])
         end;
     end;
     writeln;
end;

procedure printmatrix(m:matrix);
var r,c:integer;
begin
     for r:= 1 to noofrows(m) do
         begin
         for c:= 1 to noofcols(m) do
             write(m[r,c]:8:2);
         writeln;
         end;
end;

procedure getrow(m:matrix;rownumber:integer; var r:row);
var c:integer;
begin
     for c:= 0 to noofcols(m) do
     r[c]:=m[rownumber,c];
end;
```

```
procedure getcolumn(m:matrix; colnumber:integer; var c:column);
var r:integer;
begin
     for r:= 0 to noofrows(m) do
     c[r]:=m[r,colnumber];
end;

procedure putrow(var m:matrix; rownumber:integer; r:row);
var c:integer;
begin
     for c:= 1 to noofcols(m) do
         m[rownumber,c]:=r[c];
end;

procedure putcolumn (var m:matrix; colnumber:integer; c:column);
var r:integer;
begin
     for r:= 0 to noofrows(m) do
         m[r,colnumber]:=c[r];
end;

procedure multrow(operand:row; multiplier:extended; var result:row);
var c:integer;
begin
     for c:= 1 to trunc(operand[0]) do
         result[c]:= operand[c]*multiplier;
     result[0]:=operand[0];
end;

procedure multcol (operand:column; multiplier:extended; var result:column);
var r:integer;
begin
     for r:= 1 to trunc(operand[0]) do
         result[r]:= operand[r]*multiplier;
end;

procedure subtractrow(first,second:row; var result:row);
var c:integer;
begin
     if trunc(first[0])<>trunc(second[0])
        then writeln('Unable to subtract.....')
        else
        begin
             result[0]:=first[0];
             for c:=1 to trunc(first[0]) do
                 result[c]:=first[c]-second[c];
        end;
end;

procedure multmatrix(first,second:matrix; var result:matrix);
```

```
var r,c,m:integer;
    temp:extended;
begin
     if noofcols(first)<>noofrows(second)
     then writeln('No product exists...')
     else
     begin
          result[1,0]:=noofcols(second)/1;
          result[0,1]:=noofrows(first)/1;
               for r:= 1 to noofrows(first) do
               begin
                    for c:=1 to noofcols(second) do
                    begin
                    temp:=0;
                    for m:=1 to noofcols(first) do
                    temp:=temp+first[r,m]*second[m,c];
                    result[r,c]:=temp;
                    end;
               end;
     end;
end;

procedure makeidentity(var I:matrix; t:extended);
var r,c:integer;
begin
     for r:=1 to trunc(t) do
         for c:= 1 to trunc(t) do
              if r=c then I[r,c]:=1
                 else I[r,c]:=0;
     for r:=1 to trunc(t) do
         begin
              I[0,r]:=t;
              I[r,0]:=t;
         end;
end;

procedure printrow(r:row);
var p:integer;
begin
     for p:= 1 to trunc(r[0]) do
         write(r[p]:5:2);
     writeln
end;

procedure exchangerows(var m:matrix; r1,r2:integer);
var first,second:row;
begin
     getrow(m,r1,first);
     getrow(m,r2,second);
     putrow(m,r1,second);
```

```
     putrow(m,r2,first)
end;

function nextnonzeroincol(m:matrix; r,c:integer):integer;
var found:boolean;
begin
     found:=false;
     r:=r+1;
     while (r<noofrows(m)) and (not found) do
     begin
          if m[r,c]<>0 then
             found:=true
             else
             r:=r+1;
     end;
     if r>noofrows(m)
     then nextnonzeroincol:=0
     else nextnonzeroincol:=r
end;

procedure pause;
var c:char;
begin
write('press return key to continue....');
readln(c)
end;

function poslargestincol(c:integer;m:matrix):integer;
var temp:extended;
    i,pos:integer;
begin
     pos:=1;
     temp:=m[1,c];
     for i:=1 to noofrows(m) do
     if m[i,c]>temp then
     begin
     pos:=i;
     temp:=m[i,c]
     end;
     poslargestincol:=pos
end;

procedure arrangecols(var m,n:matrix);
var temp:integer;
begin
     temp:=poslargestincol(1,m);
     if temp<>1 then
     begin
          exchangerows(m,1,temp);
          exchangerows(n,1,temp);
```

```
     end;
end;

procedure GaussInvert(m:matrix; var I:matrix);
var singular:boolean;
    r,c,tempint:integer;
    factor:extended;
    D:array[1..Maxcol] of extended;
    mrow,Irow,Icurrent,mcurrent:row;
begin
     singular:=false;
     makeidentity(I,m[0,1]);
     for c:= 1 to noofcols(m) do
     begin
         if m[c,c]=0.00000000 then
         begin
              tempint:=nextnonzeroincol(m,c,c);
              if tempint>0 then
              begin
                   exchangerows(m,tempint,c);
                   exchangerows(I,tempint,c)
              end
              else
              begin
              writeln('Matrix is singular. No Inverse exists');
              singular:=true;
              end;
         end;
         if not singular then
         begin
         for r:= 1 to noofrows(m) do
                  begin
                  if (r<>c) and (m[r,c]<>0.0000000000) then
                  begin
                       factor:=m[c,c]/m[r,c];
                       getrow(m,r,mrow);
                       getrow(I,r,Irow);
                       getrow(m,c,mcurrent);
                       getrow(I,c,Icurrent);
                       multrow(mrow,factor,mrow);
                       multrow(Irow,factor,Irow);
                       subtractrow(mrow,mcurrent,mrow);
                       subtractrow(Irow,Icurrent,Irow);
                       putrow(I,r,Irow);
                       putrow(m,r,mrow);
                   end;
                   end;
                   end;
                   end;
         if not singular then
```

```
          for r:= 1 to noofrows(I) do
          begin
               factor:=m[r,r];
               for c:= 1 to  noofcols(I) do
                       I[r,c]:=I[r,c]/factor;
          end;
end;
```