```
%%%%%%%%%%%%%%%%%%%%
% es 1
A=[ 1 2 3 4;
 -1 2 5 -2;
2 0 4 -3;
10 -1 -3 0;
1 1 1 -1];
[n,m]=size(A);

[Q1,R1]=gram(A);

disp(' ')
fprintf('Ortogonalizzazione di Gram-Schmidt\n')
fprintf('   ||I-Q^TQ||    ||A-QR||    cond(A)   cond(R)\n')
disp([norm(eye(m)-Q1'*Q1),norm(A-Q1*R1),cond(A),cond(R1)])

% es 1 punto iii)
Atilde=[  1     2    -5     4
    -1     2    -7    -2
     2     0     2    -3
    10    -1    13     0
     1     1    -2    -1 ];
[n,m]=size(Atilde);

[Q2,R2,loss_rank]=gramrr(Atilde);
disp(' ')
fprintf('Ortogonalizzazione di Gram-Schmidt, rank reveiling:\n')
fprintf('Indice colonne linearmente dipendenti:\n')
disp(loss_rank)
dim_im=size(Atilde,2)-length(loss_rank);
fprintf('dimensione immagine di A: %d \n', dim_im)

Q2true=Q2; Q2true(:,loss_rank)=[]; R2true=R2; R2true(loss_rank,:)=[];
fprintf('   ||I-Q^TQ||    ||A-QR||    cond(A)   cond(R)\n')
disp([norm(eye(dim_im)-Q2true'*Q2true),norm(Atilde-
Q2true*R2true),cond(Atilde),cond(R2true)])

%%%%%%%%%%%%%%%%%%%%
% es 2
n=10;m=6;
A=randn(n,m);
[Q,R]=QR_house(A);
disp(' ')
fprintf('Ortogonalizzazione di Householder \n')
fprintf('   ||I-Q^TQ||    ||A-QR||    cond(A)   cond(R)\n')
disp([norm(eye(n)-Q'*Q),norm(A-Q*R),cond(A),cond(R)])

%%%%%%%%%%%%%%%%%%%%
% es 3 (facoltativo)
A=[ 1 2 3 4;
 -4 -1 5 -2;
0 3 4 -3;
0 0 -3 0;
0 0 0 -1];
[n,m]=size(A);
[Q,R]=QR_givens(A);
disp(' ')
fprintf('Ortogonalizzazione di Givens \n')
fprintf('   ||I-Q^TQ||    ||A-QR||    cond(A)   cond(R)\n')
disp([norm(eye(n)-Q'*Q),norm(A-Q*R),cond(A),cond(R)])



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% CUT HERE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Q,R]=gram(A)
%function [Q,R]=gram(A)
%
[n,m]=size(A);

R(1,1)=norm(A(1:n,1)); Q(:,1)=A(:,1)/R(1,1);
for i=2:m,

  R(1:i-1,i)=Q(1:n,1:i-1)'*A(1:n,i);
  q1=A(1:n,i) - Q(1:n,1:i-1)*R(1:i-1,i);
  R(i,i)=norm(q1);
  if R(i,i) < 1e-14
       fprintf('linear dependence\n'); break
  else
    Q(1:n,i)=q1/R(i,i);
  end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% CUT HERE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Q,R,loss_rank]=gramrr(A)
%function [Q,R,loss_rank]=gramrr(A)
%
[n,m]=size(A);
loss_rank=[];

R(1,1)=norm(A(1:n,1)); Q(:,1)=A(:,1)/R(1,1);
for i=2:m,

  R(1:i-1,i)=Q(1:n,1:i-1)'*A(1:n,i);
  q1=A(1:n,i) - Q(1:n,1:i-1)*R(1:i-1,i);
  R(i,i)=norm(q1);
  if R(i,i) < 1e-14
       fprintf('linear dependence. column %d\n', i);
       Q(1:n,i)=0*q1;  loss_rank=[loss_rank,i];
  else
    Q(1:n,i)=q1/R(i,i);
  end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% CUT HERE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Q,R]=QR_house(A)

[n,m]=size(A);
R=A;
U=eye(n);

for k=1:m
   x=R(k:n,k);
   alpha = - sign(x(1))*norm(x);
   e1=eye(n-k+1,1);
   v=x-alpha*e1;
   beta = 2/(v'*v);
   R(k:n,k:m)=R(k:n,k:m) - v*(beta*(v'*R(k:n,k:m)));
   U(k:n,1:n)=U(k:n,1:n) - v*(beta*(v'*U(k:n,1:n)));
 end
Q=U';
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% CUT HERE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Q,R]=QR_givens(A)

[n,m]=size(A);
R=A;
U=eye(n);

for k=1:m
   x=R(k:k+1,k);
   normx=norm(x);
   c=x(1)/normx; s=x(2)/normx;
   G=[c s;-s c];
   R(k:k+1,k:m)=G*R(k:k+1,k:m);
   U(k:k+1,1:n)=G*U(k:k+1,1:n);
 end
Q=U';
end
```