```
program questao1
  implicit none
  real, dimension(1000) :: x, y
  integer :: n, i
  real :: a, b, h, integra

  open(file='dados.dat', unit=1, status='old')

  read (1, *) n

  do i = 1, n
    read(1,*) x(i), y(i)
  end do

  print *, "O valor do deslocamento médio do automóvel é:"
  print *, integra(n, x, y)


end program questao1

function integra(n, x, y) result(result)
  implicit none
  real, dimension(1000) :: x, y
  real :: a, b, h, aux
  real :: result
  integer :: i, n

  a = x(1)
  b = x(n)
  h = (b-a)/n

  aux = 0.0
  do i = 2, (n-1)
    aux = aux+y(i)
  end do

  result = (h/2)*(y(1)+2*aux+y(n))
end function integra

```

```
program questao2
  implicit none
  real, dimension(3, 3) :: A
  real, dimension(3) :: b, x
  integer :: n, i

  n = 3

  data (A(1,i), i=1,3) /  3.0,  2.0,  -1.0 /
  data (A(2,i), i=1,3) /  1.0,  3.0,  1.0 /
  data (A(3,i), i=1,3) /  2.0,  2.0,  -2.0 /

  data (b(i), i=1,3) /  0.0,  1.0,  2.0 /

  call jordan(n, A, b, x)

  print *, "Resultado do sistema de equações lineares"
  do i = 1, 3
    print *, x(i)
  end do
end program questao2

subroutine jordan(n, A, b, x)
  implicit none
  real, dimension(3, 3) :: A
  real, dimension(3) :: b, x
  real :: m
  integer :: i, j, k, n

  do k = 1, n
    do i = 1, n
      if ( i .ne. k ) then
        m = A(i, k)/A(k, k)
        A(i, k) = 0.0
        do j = k+1, n
          A(i, j) = A(i, j) - m*A(k, j)
        end do
        b(i) = b(i) - m*b(k)
      end if
    end do
  end do

  do i = 1, n
    x(i) = b(i)/A(i, i)
  end do
end subroutine jordan
```

```
program questao3
  implicit none
  real :: V, L, R, i0, t0, i, t, tf, step
  real, external :: didt

  t0 = 0
  i0 = 0

  t = t0
  i = i0

  tf = 1
  step = 1E-3


  print *, "Insira o valor da tensão CC"
  read(*,*) V
  print *, "Insira o valor da indutância do indutor"
  read(*,*) L
  print *, "Insira o valor da resistência elétrica"
  read(*,*) R

  call euler(didt, V, L, R, i, t, tf, step)

  print *, "O valor da corrente elétrica no tempo final:"
  print *, i

end program questao3

subroutine euler(didt, V, L, R, i, t, tf, step)
  implicit none
  real :: didt, V, L, R, i, t, tf, step

  do while(t<tf)
    i = didt(V, L, R, i, t)*step + i
    t = t + step
  end do

end subroutine euler

function didt(V, L, R, i, t) result(result)
  implicit none
  real :: V, L, R, i, t
  real :: result

  result = (V-R*i)/L

end function didt
```