```
loadPackage "Dmodules"
R = QQ[x][Dx]

discriminantTest = L -> (
 if not instance(L, D) then error "L must be a differential operator in the ring D";
 I := indicialPolynomial(L, x);
 Lform := listForm I;
 c2 := 0;
 c1 := 0;
 c0 := 0;
 scan(Lform, term -> (
  if term#1 == 2 then c2 = term#0;
  if term#1 == 1 then c1 = term#0;
  if term#1 == 0 then c0 = term#0;
 ));
 if c2 == 0 then error "Operator not second-order Euler-type";
 Delta := c1^2 - 4*c2*c0;
 eps := 1e-12;
 if Delta < -eps then (
  alpha := -c1/(2*c2);
  beta := sqrt(-Delta)/(2*c2);
  basis := {x^alpha*cos(beta*log(x)), x^alpha*sin(beta*log(x))};
  classif := "oscillatory";
  t := first timing S := Dsolve(L);
  return hashTable {"classification" => classif, "discriminant" => Delta, "basis" => basis, "solution"
=> S, "time_s" => t}
```

```
 ) else if Delta > eps then (
  r1 := (-c1 + sqrt(Delta))/(2*c2);
  r2 := (-c1 - sqrt(Delta))/(2*c2);
  basis := {x^r1, x^r2};
  classif := "real-distinct";
  t := first timing S := Dsolve(L);
  return hashTable {"classification" => classif, "discriminant" => Delta, "basis" => basis, "solution"
=> S, "time_s" => t}
 ) else (
  rho0 := -c1/(2*c2);
  basis := {x^rho0, x^rho0*log(x)};
  classif := "real-repeated";
  t := first timing S := Dsolve(L);
  return hashTable {"classification" => classif, "discriminant" => Delta, "basis" => basis, "solution"
=> S, "time_s" => t}
 )
)
```