```
(*Jacobi Method*)

(*Change this to whatever you want. this determines the size
 of our nxn matrix. Will use small matrix as proof of concept*)
size = 10;

(*create matrix A*)
A = ConstantArray[0, {size, size}];

min = 1;
max = 9;

(*creating a strictly positive matrix*)
For[ i = 1, i ≤ size, i++,
  For[j = 1, j ≤ size, j++,
    If[i ≤ j, A[[i, j]] = RandomInteger[{min, max}]];
   ];
 ];

For[ i = 1, i ≤ size, i++,
  For[j = 1, j ≤ size, j++,
    If[i ≥  j, A[[i, j]] = A[[j, i]];
     ];
   ];
 ];

(*adding up the columsn of each row to force diuagonal dominance*)
sums = Total[A];

(*enforcing diagonal dominance on matrix A*)
For[i = 1, i ≤ size, i++,
 A[[i, i]] = RandomInteger[{sums[[i]], sums[[i]] + 1}];
]
(*making vector b*)
b = ConstantArray[0, {Length[A], 1}];

(*making the vector b to be <1,2,3,4,5,6,7,8,9,10>*)
For[i = 1, i ≤ Length[A], i++,
  b[[i]] = i;
 ];

(*S is my matrix of the diagonals of matrix A,
and it's called S because mathematica reserves the variable D for some reason... *)
S = ConstantArray[0, {Length[A], Length[A]}];

(*R is my matrix of everything of A, except the diagonal, where there value is 0*)
R = ConstantArray[0, {Length[A], Length[A]}];

(*actually making whatever I just said above*)
For[m = 1, m ≤ Length[A], m++,
  For[n = 1, n ≤ Length[A] , n++,
    If[m == n, S[[m, n]] = A[[m, n]], R[[m, n]] = A[[m, n]];
    ]
   ]
  ];

(*choosing x1 = a vector of all 1's because I can...*)

x_1 = ConstantArray[1, {Length[A], 1}];

(*iteration process of jacobi method, done 5000 times*)
For[i = 1, i ≤ 1000, i++,
  x_{i+1} = Inverse[S].(b - R.x_i);
  last = x_{i+1};
 ];
(*showing the structure of the matrices and vectors we're working with*)
MatrixForm[A]
MatrixForm[S]
MatrixForm[R]
MatrixForm[b]

xreal = LinearSolve[A, b]; (*since we chose a small system, this is the actual *)
xbar = last; (*the 1000's iteration of the iterated x*)
b1 = A.xbar; (*creating vector b1, which should approximate vector b*)
precision = 30; (*going to numerically round to the nearest 30th decimal*)
Print[MatrixForm[N[b1, precision]]]; (*if b1 ~ b,
then we're correct, b1 should be close to b = <1,2,3,4,5,6,7,8,9,10*)
```