```
matrix_function.soltion: Method in matrix_function that
        do a progressive or backward substitutionv to find an array

Input: matrix A, array b, int size
Output: solution vector results

begin doolittle

L = identityMatrix(size)
U = identityMatrix(size)

int count <- 0
while (count < size) do
        int count2 <- count
        while (count2 < size) do
                float sum <- 0
                int count3 <- 0
                while (count3 < count) do
                        sum <- sum + (L[count][count3]*U[count3][count2])
                end while
                U[count][count2] <- A[count][count2]-sum
        while (count2 < size) do
                if (count == count2):
                        L[conut][count] -<- 1
                else:
                        sum <- 0
                        while (count3 < count) do
                                sum <- sum + (L[count2][count3]*
                                    U[count3][count])
                        end while
                        L[count2][count] <- ((A[count2][count]-sum)/
                            U[count][count]
z = array(matrix_function.soltion(L,b))
x = matrix_function.soltion(U,z)

array sol
int count <- 0
while (count < size(x)) do
        sol[count] <- x[i]
return sol

end doolittle
```