Comment:
dictionary data is a dictionarity (json format) that have
every Hi Hermite coefficient and have the final polynomial
sm is a import of sympy from python to to represent variables
in a polynomial

parse_expr(): this function that change a string to sympy expression
(is a function from sympy)
lagrange.lagrange: its a methos in the lagrange.py file that give us
the lagrange coefficient

diff(x) it's a method from sympy to found a derivative of a specific
function

json.dumps it's a method to do a json from a dict

Input: array arrayX array arrayY, array arrayz, int size
Output: json data

```
begin hermite
x <- sm.symbols('x')
array arrayAux
array arrayDerivate
array arraySquare
array H
array H2
int counter <- 0
dictionarity dic <- lagrange.lagrange(x,y,size)
while (counter < size(dic)-1) do:
        arrayAux.add(parse_expr(dic[counter))
        arraySquare.add(parse_expr(dic[counter)*parse_expr(dic[counter))
        counter <- counter + 1
counter <- 0
while (counter < size(arrayAux)) do:
        arrayDerivate.add(arrayAux[counter].diff(x))
        couunter <- counter + 1
counter <- 0
while (counter < size) do:
        value <- arrayDerivate[counter].subs(x,arrayX)
        aux <- (((x-arrayX[counter])*value*(-2))+1)
        aux <- aux * arraySquare[counter]
        H.add(aux)
        value <- (x-arrayX[i])*arraySquare[i]
    H2.add(value)
        counter <- counter + 1
```

```
polynomial <- 0
counter <- 0
while (counter < size(H)) do:
        results[counter] <- arrayY[counter]*H[counter]
        polynomial <- polynomial+(arrayY[counter]*H[counter])
        counter <- counter + 1
results["polynomial"] <- polynomial
data <- json.dumps(results)
return data
end hermite
```