```
(*Parameters*)

PI = N[π, 12];
EPS = 1.*^-12;

(*Pauli operators*)

σI = {{1., 0.}, {0., 1.}};
σX = {{0., 1.}, {1., 0.}};
σY = {{0., -1. I}, {1. I, 0.}};
σZ = {{1., 0.}, {0., -1.}};

funPX[i_] := Module[{PX}, (
    If[i == 0, PX = σX, PX = σI];
    Do[(
        If[i == k, PX = KroneckerProduct[σX, PX], PX = KroneckerProduct[σI, PX]];
    ), {k, 1, Nq - 1}];
    Return[PX]
)]

funPY[i_] := Module[{PY}, (
    If[i == 0, PY = σY, PY = σI];
    Do[(
        If[i == k, PY = KroneckerProduct[σY, PY], PY = KroneckerProduct[σI, PY]];
    ), {k, 1, Nq - 1}];
    Return[PY]
)]

funPZ[i_] := Module[{PZ}, (
    If[i == 0, PZ = σZ, PZ = σI];
    Do[(
        If[i == k, PZ = KroneckerProduct[σZ, PZ], PZ = KroneckerProduct[σI, PZ]];
    ), {k, 1, Nq - 1}];
    Return[PZ]
)]

funPO[ps_] := Module[{q, p, PO}, (
    q = 0;
    p = StringTake[ps, {q + 1}];
    If[StringMatchQ[p, "I"], PO = σI];
    If[StringMatchQ[p, "X"], PO = σX];
    If[StringMatchQ[p, "Y"], PO = σY];
    If[StringMatchQ[p, "Z"], PO = σZ];
    Do[(
        p = StringTake[ps, {q + 1}];
        If[StringMatchQ[p, "I"], PO = KroneckerProduct[σI, PO]];
        If[StringMatchQ[p, "X"], PO = KroneckerProduct[σX, PO]];
        If[StringMatchQ[p, "Y"], PO = KroneckerProduct[σY, PO]];
        If[StringMatchQ[p, "Z"], PO = KroneckerProduct[σZ, PO]];
    ), {q, 1, Nq - 1}];
    Return[PO]
)]
```

```
(*Hamiltonian*)

funHamiltonianQubit[Model_] := Module[{Ham}, (
    Ham = 0.;
    Do[(
        Ham = Ham + Model[[i, 1]] * funPO[Model[[i, 2]]];
    ), {i, 1, Length[Model]}];
    Return[Ham]
)]

(*Spectrum*)

funSpectrum[Ham_] := Module[{vals, vecs}, (
    {vals, vecs} = Eigensystem[Ham];
    vals = Re[vals];
    {vals, vecs} = Transpose@SortBy[Transpose[{vals, vecs}], First];
    (*Print[Total[
      Total[Abs[Transpose[vecs].DiagonalMatrix[vals].Conjugate[vecs]-Ham]]]];*)
    (*Print[Total[
      Total[Abs[DiagonalMatrix[vals]-Conjugate[vecs].Ham.Transpose[vecs]]]]];*)
    Return[{vals, vecs}]
)]
```