QUES 1.

Given any function F(x), write a code to obtain the Fourier Series.

SOLUTION:

ALGORITHM:

```
1. Input no. of terms in Fourier Series say n.
2. Create a zeros matrix for $a_n$ and $b_n$ and set $a_0$ to zero. Then Input number of discontinuities
3. If (discontinuity ~= 0)
    • Input lower and upper limit of whole function.
    • Start for loop with loop(variable) from zero to discontinuity.
        ➢ Input function and its upper and lower limit discretely
        ➢ Start for loop with k(variable) from 1 to n
            ▪ F= (entered function) *cos(k*x)
            ▪ $a_n$ (k)= $a_n$ (k) + invoke function -> integral(@(x) eval(vectorize(f)), lower
              limit of discrete function, upper limit of discrete function).
            ▪ Repeat above steps for $b_n$ with F= (entered function) * sin(k*x)
        ➢ $a_0$ = $a_0$ + invoke function -> integral(@(x) eval(vectorize(entered function)),
          lower limit of discrete function,  upper limit of discrete function).
        ➢ Start for loop with k=1: n
            ▪ Set $a_n$(k)=$a_n$(k)*2/(upper-lower limit of whole function)
            ▪ Repeat above step by replacing $a_n$(k) with $b_n$(k)
4. else Input function lower and upper limit of whole function
    • Start for loop with k(variable) from 1 to n
        ➢ F= (entered function) *cos(k*x)
        ➢ $a_n$ (k)= $a_n$ (k) + (2/(upper – lower limit))* integral(@(x) eval(vectorize(f)), lower
          limit of function, upper limit of function).
        ➢ Repeat above steps for $b_n$ with F= (entered function) * sin(k*x)
    • $a_0$ = $a_0$ + invoke function -> integral(@(x) eval(vectorize(entered function)), lower limit of
      discrete function,  upper limit of discrete function).
5. Print $a_n$, $b_n$, $a_0$.
```

QUES 2.

Given any function F(x), write a code to check its HURWITZNESS.

SOLUTION:

ALGORITHM:

```
1. Input coefficient matrix of function and store it in any variable, say p.
2. Set any two variables say q and g initially to one.
3. Invoke function roots(), with parameter as p and store roots in a variable, say r.
4. Start for loop with k starting from 1 to length(p).
       ➢ Check if(real(p(k))~=p(k) ), set any variable, say g, to zero and then break.
5. Start for loop with k starting from 1 to length(r)
       ➢ Check if(real(r(k))>10^(-15)), set any variable, say q, to zero and then break.
6. Check if(g==1)
       ➢ Print out 'Function is real for real s'
       ➢ Else
             o Print out 'Function is not real for real s'
7. Check if(q==0 and g==1)
       ➢ Print out 'non Hurwitz'
       ➢ Else if(q==1 and g==1)
             o Print out 'Hurwitz'
8. Input Y/N for showing roots of the given polynomial
       ➢ If(input == 'y' or 'Y')
             o Start a for loop with k=1 through length(r) and print ' r(k)'
```

QUES 3.

Given any function F(s)=N(s)/D(s), write a code to check if given function is PRF or not.

SOLUTION:

Conditions that would be checked are

```
1.Z(s) is real for real s
2.(N(s)+D(s)) is HURWITZ
3.D(s) is HURWITZ
4.Poles on Imaginary Axis are SIMPLE
5.Residue of poles on Imaginary Axis are real and positive
6.Residue of poles on Imaginary Axis are complex conjugate (two pure reals
are also considered complex conjugate)
7.Re(Z(jw))>0 for all w'
```

ALGORITHM:

1. Input numerator and denominator coefficient.
2. Save numerator and denominator coefficient in other variables, say n2 and d2.
3. Make both the matrices of equal length
   - If(length(numerator)>length(denominator))
     - Assign values of denominator to a temporary variable and make denominator matrix equal to zero using for loop.
     - Start a for loop with variable k from 1 through length(d)
       - d(length(numerator)-length(temp variable)+k)=temp(k)
   - Repeat similar steps for length(numerator)>length(denominator)
4. Assign roots of denominator to a variable say den, assign a variable, say z the value of numerator + denominator and roots of z to a variable, say r.
5. Start a for loop with k starting from 1 through length(den)
   - Check if(real(den(k))>0) set a variable, say denomi =1
   - Check if(real(r(k))>-0.000000000000001) set a variable, say q=1
6. Start a for loop with k starting from 1 through length(z)
   - Check if(real(z(k)) is not equal to z(k)) set a variable, say l=1
7. Invoke function residue() with parameters n and d and store it in variables [r1 p1 k1]
8. Start a for loop with k=1 through length of p1
   - Check if(real(p1(k))==0)
     - Start a for loop with ch=k+1 through length(p1)
       - Check if(p1(k)==p1(ch)) set a variable, say answer=1
     - Again start a for loop with ch=1 through k-1
       - Check if(p1(k)==p1(ch)) again set answer=1
9. Start a for loop with k=1 through length(p1)
   - Check if(real(p1(k))==0)
     - Check if(imag(p1(k))~=0 or real(r1(k))<0.0000000000000001)
       - Set a variable, say answer2=1 and break out of loop
10. Set a variable, say answer3=0 and then Start a for loop with k=1 through length(p1)
    - Check if(real(p1(k))==0)
      - Start a for loop with check=1 through length(r1)
        - Check if(imag(r1(k))==-imag(r1(check))&&real(r1(k))==real(r1(check))) set a variable, say answer3=answer3+1

```
11. if(l==0) print “polynomial passed test one” else print ”polynomial failed test one ”
12. if(q==0) print “polynomial passed test two” else print ”polynomial failed test two ”
13. if(denomi==0) print “polynomial passed test three” else print ”polynomial failed test three ”
14. if(answer==0) print “polynomial passed test four” else print ”polynomial failed test four
15. if(answer2==0) print “polynomial passed test five” else print ”polynomial failed test five
16. if((call function rem() with parameters answer3 and 2)==0) print “polynomial passed test six” else
    print ”polynomial failed test six”
17. Start a for loop with k=1 through length(n2)
        ▪ n2(k)=n2(k)/n2(1)
18. Start a for loop with k=1 through length(d2)
        ▪ d2(k)=d2(k)/d2(1)
19. check if(length(n2)<4 && length(d2)<4)
        ▪ check if(length(n2)==2&&length(d2)==3)
            ➢ check if(n2(2)>0&&d2(2)>0&&d2(3)>0&&d2(2)>=n2(2)) print “polynomial
              passed test 7”
            ➢ else if(length(n2)==3&&length(d2)==3)
                o check if (n2(2)*d2(2)>= (sqrt(n2(3))-sqrt(d2(3)))^2) print “polynomial passed
                  test 7 ”
20. else print “Condition 7 can only be checked for linear/quadratic or quadratic/quadratic rational
    functions ”
```