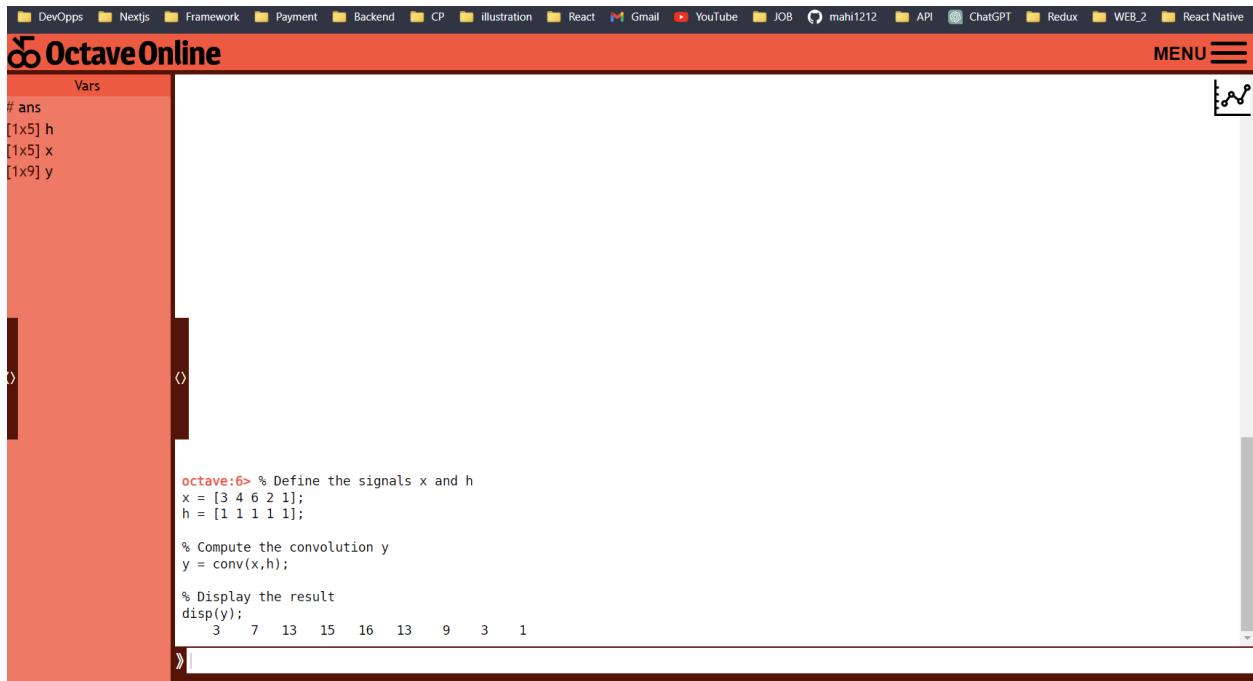


ANS TO THE QUESTION NUMBER: 1

CODE:

```
x = [3 4 6 2 1];  
h = [1 1 1 1 1];  
  
% Compute the convolution y  
y = conv(x,h);  
disp(y);
```



The screenshot shows the OctaveOnline web interface. At the top, there is a navigation bar with various icons and labels like 'DevOps', 'Nextjs', 'Framework', 'Payment', 'Backend', 'CP', 'illustration', 'React', 'Gmail', 'YouTube', 'JOB', 'mahi1212', 'API', 'ChatGPT', 'Redux', 'WEB_2', and 'React Native'. Below this, the 'Vars' panel on the left lists the variables: '# ans', '[1x5] h', '[1x5] x', and '[1x9] y'. The main editor area contains the MATLAB code from the previous block. The output of the code is displayed at the bottom of the editor: '3 7 13 15 16 13 9 3 1'. A small plot icon is visible in the top right corner of the editor area.

ANS TO THE QUESTION NUMBER: 3

Code: % Create a time vector

```
t = linspace(0, 5, 100);
```

% Define functions for unit step, unit ramp, unit parabolic, and exponential signals

```
unit_step = heaviside(t);
```

```
unit_ramp = t;
```

```
unit_parabolic = t.^2;
```

```
exponential = exp(t);
```

```
% Create a 4x4 subplot grid
```

```
subplot(2, 2, 1);
```

```
plot(t, unit_step, 'b');
```

```
title('Unit Step');
```

```
xlabel('Time');
```

```
ylabel('Amplitude');
```

```
subplot(2, 2, 2);
```

```
plot(t, unit_ramp, 'r');
```

```
title('Unit Ramp');
```

```
xlabel('Time');
```

```
ylabel('Amplitude');
```

```
subplot(2, 2, 3);
```

```
plot(t, unit_parabolic, 'g');
```

```
title('Unit Parabolic');
```

```
xlabel('Time');
```

```
ylabel('Amplitude');
```

```
subplot(2, 2, 4);
```

```
plot(t, exponential, 'm');
```

```
title('Exponential');
```

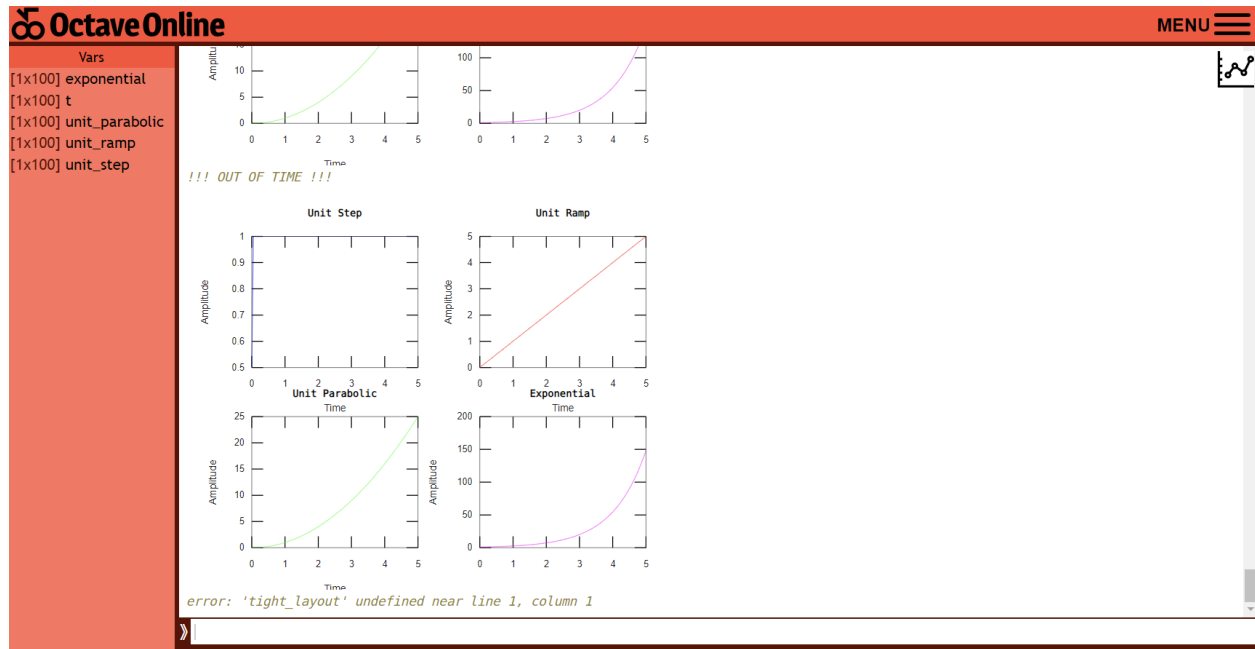
```
xlabel('Time');
```

```
ylabel('Amplitude');
```

```
% Adjust layout for better spacing
```


```
tight_layout();
```


Output:



ANS TO THE QUESTION NUMBER: 4

Output:



MENU 

Vars

[1x4] h

k

len_h

len_x

n

[1x4] x

[1x7] y

```

octave:113> % Given impulse response and input
h = [1, 1, 1, 1];
x = [5, 7, 1, 4];

% Length of the signals
len_h = length(h);
len_x = length(x);

% Initialize the output signal y
y = zeros(1, len_h + len_x - 1);

% Convolution sum
for n = 1:len_h + len_x - 1
    for k = max(1, n - len_h + 1):min(len_x, n)
        y(n) = y(n) + x(k) * h(n - k + 1);
    end
end

% Display the result
disp('Impulse response h(n):');
disp(h);
disp('Input signal x(n):');
disp(x);
disp('Output signal y(n) after convolution:');
disp(y);
Impulse response h(n):
    1    1    1    1
Input signal x(n):
    5    7    1    4
Output signal y(n) after convolution:
    5   12   13   17   12    5    4

```

ANS TO THE QUESTION NUMBER: 2

Code:

% Define the Laplace variable

$s = tf('s');$

% Define the frequency of the cosine signal

$k = 2;$

% Laplace transform of $\cos(kt)$

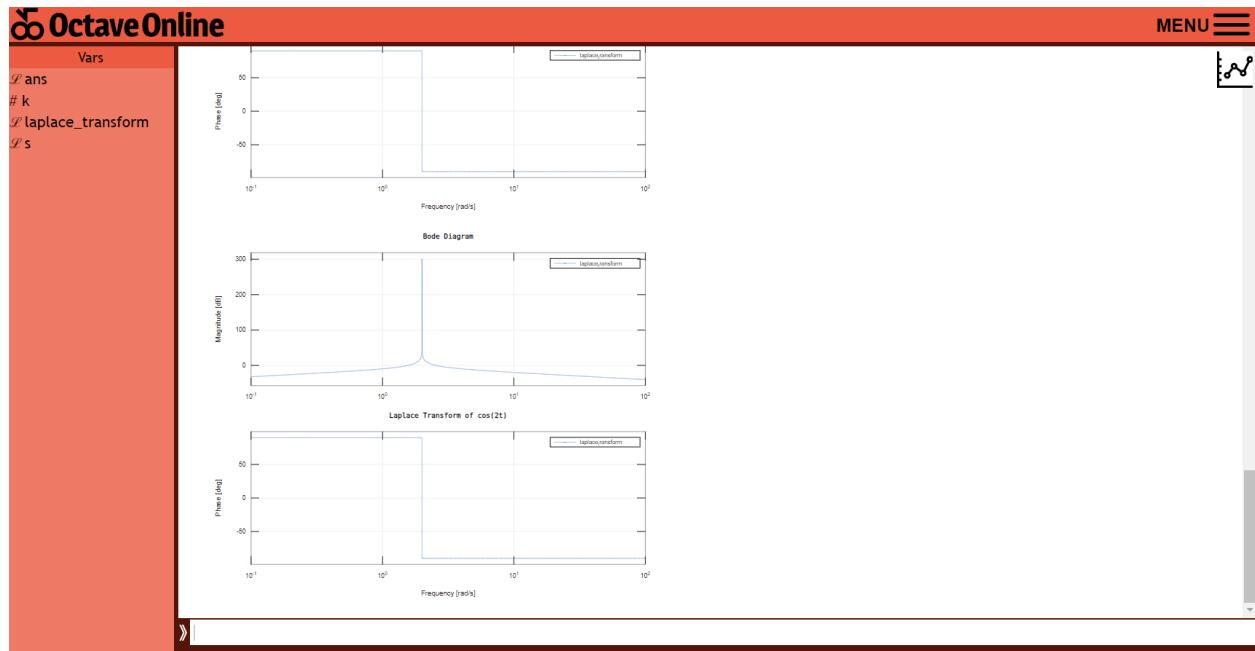
$\text{laplace_transform} = s / (s^2 + k^2);$

% Plot the Laplace transform

$\text{bode}(\text{laplace_transform});$

$\text{title}('Laplace Transform of \cos(2t)');$

Output:



ANS TO THE QUESTION NUMBER: 5

Code:

```
f = @(x) x.^3 .* exp(7*x);
```

```
% Define the range and sampling points
```

```
x_values = linspace(-10, 10, 1000);
```

```
delta_x = x_values(2) - x_values(1);
```

```
% Compute Fourier transform using FFT
```

```
N = length(x_values);
```

```
w_values = fftshift(2 * pi * linspace(-1, 1, N) / delta_x);
```

```
F = fftshift(fft(f(x_values), N) * delta_x);
```

```

% Compute inverse Fourier transform using IFFT
t_values = x_values;
f_inv = ifft(fftshift(F) / (2 * pi) * (2 * pi / delta_x), N) * N;

% Display the results
figure;

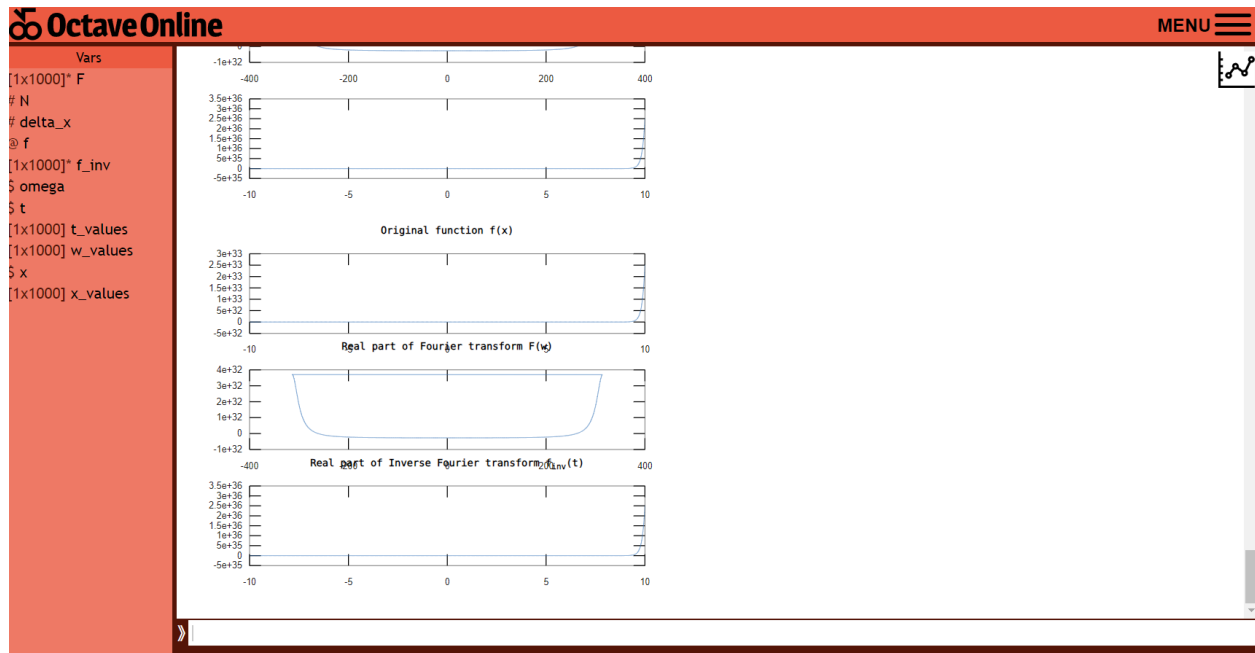
subplot(3, 1, 1);
plot(x_values, f(x_values));
title('Original function f(x)');

subplot(3, 1, 2);
plot(w_values, real(F));
title('Real part of Fourier transform F(w)');

subplot(3, 1, 3);
plot(t_values, real(f_inv));
title('Real part of Inverse Fourier transform f_{inv}(t)');

```

Output:



ANS TO THE QUESTION NUMBER: 6

Code:

% Assume the expression

`a = 0.5;`

`Xz = @(z) 1 / (1 - a * z^(-1));`

% Z-transform

`z_values = linspace(0, 1, 10); % Define Z values`

`Xz_transformed = arrayfun(@(z) Xz(z), z_values);`

% Inverse Z-transform (residue method)

`n = 0:9; % Define time indices`

`xn = a.^n; % Inverse Z-transform for this simple case`

```
% Display the results

disp('Original Expression X(z):');

disp(['1 / (1 - ' num2str(a) 'z^(-1))']);

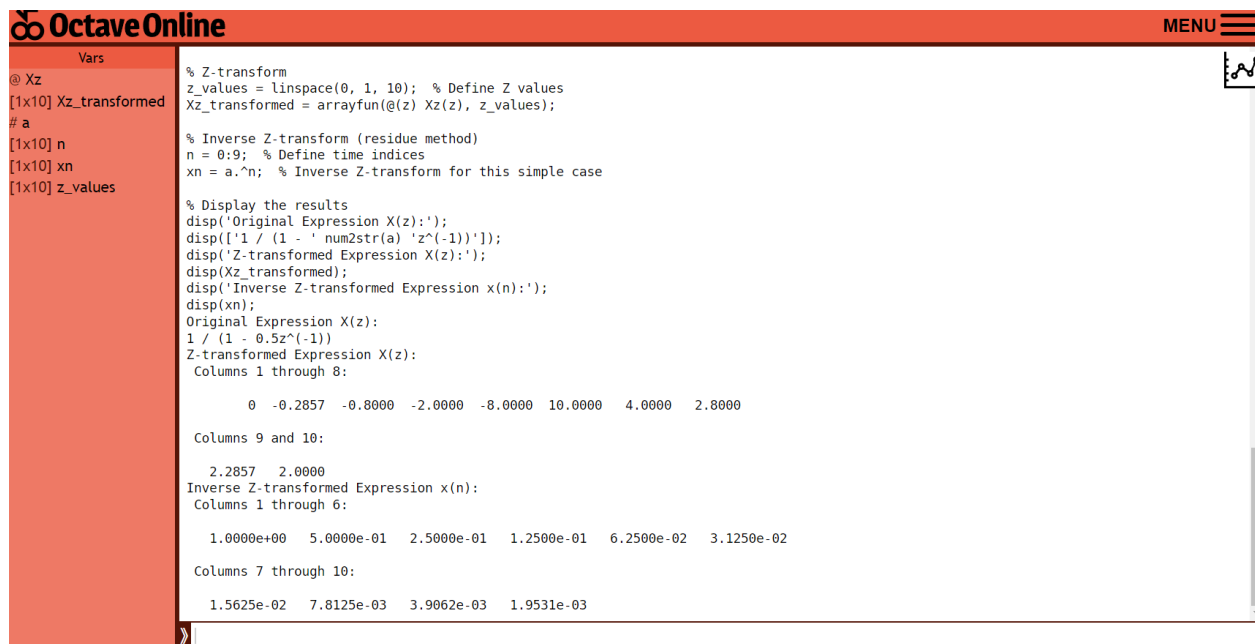
disp('Z-transformed Expression X(z):');

disp(Xz_transformed);

disp('Inverse Z-transformed Expression x(n):');

disp(xn);
```

Output:



The screenshot shows the OctaveOnline web interface. On the left, a sidebar lists variables: @ Xz, [1x10] Xz_transformed, # a, [1x10] n, [1x10] xn, and [1x10] z_values. The main area displays the following code and its output:

```
% Z-transform
z_values = linspace(0, 1, 10); % Define Z values
Xz_transformed = arrayfun(@(z) Xz(z), z_values);

% Inverse Z-transform (residue method)
n = 0:9; % Define time indices
xn = a.^n; % Inverse Z-transform for this simple case

% Display the results
disp('Original Expression X(z):');
disp(['1 / (1 - ' num2str(a) 'z^(-1))']);
disp('Z-transformed Expression X(z):');
disp(Xz_transformed);
disp('Inverse Z-transformed Expression x(n):');
disp(xn);
```

Original Expression X(z):
 $1 / (1 - 0.5z^{(-1)})$
 Z-transformed Expression X(z):
 Columns 1 through 8:
 0 -0.2857 -0.8000 -2.0000 -8.0000 10.0000 4.0000 2.8000
 Columns 9 and 10:
 2.2857 2.0000
 Inverse Z-transformed Expression x(n):
 Columns 1 through 6:
 1.0000e+00 5.0000e-01 2.5000e-01 1.2500e-01 6.2500e-02 3.1250e-02
 Columns 7 through 10:
 1.5625e-02 7.8125e-03 3.9062e-03 1.9531e-03