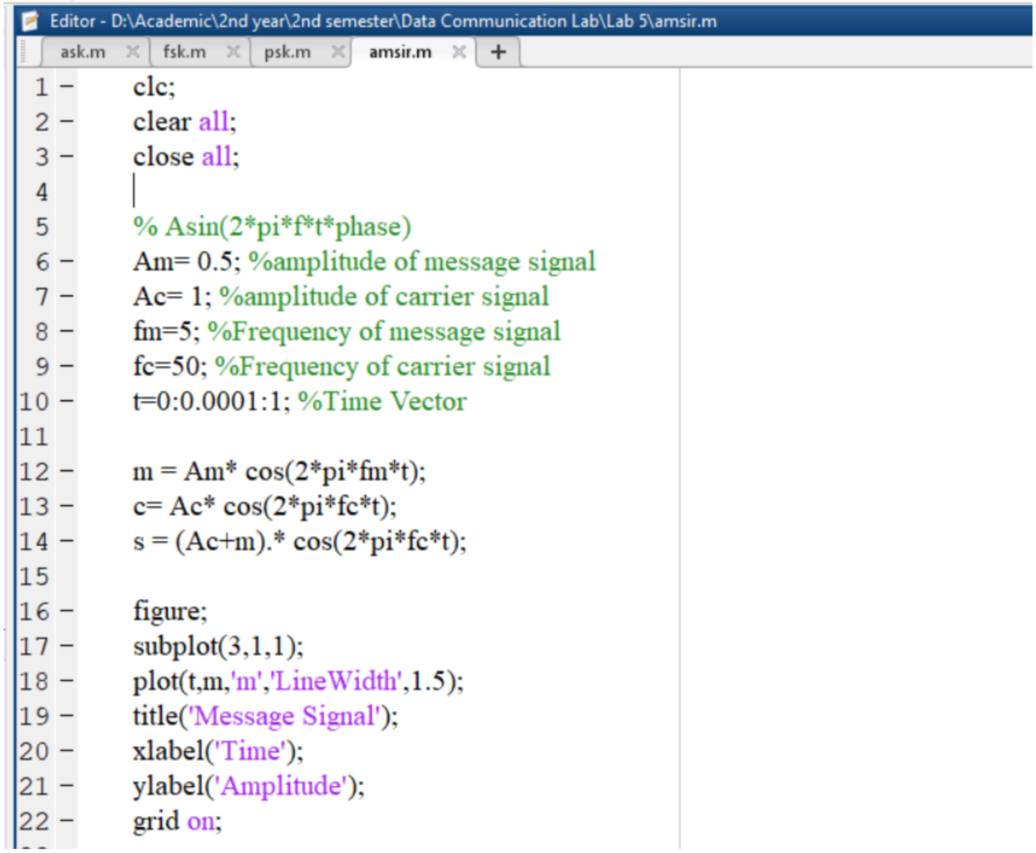
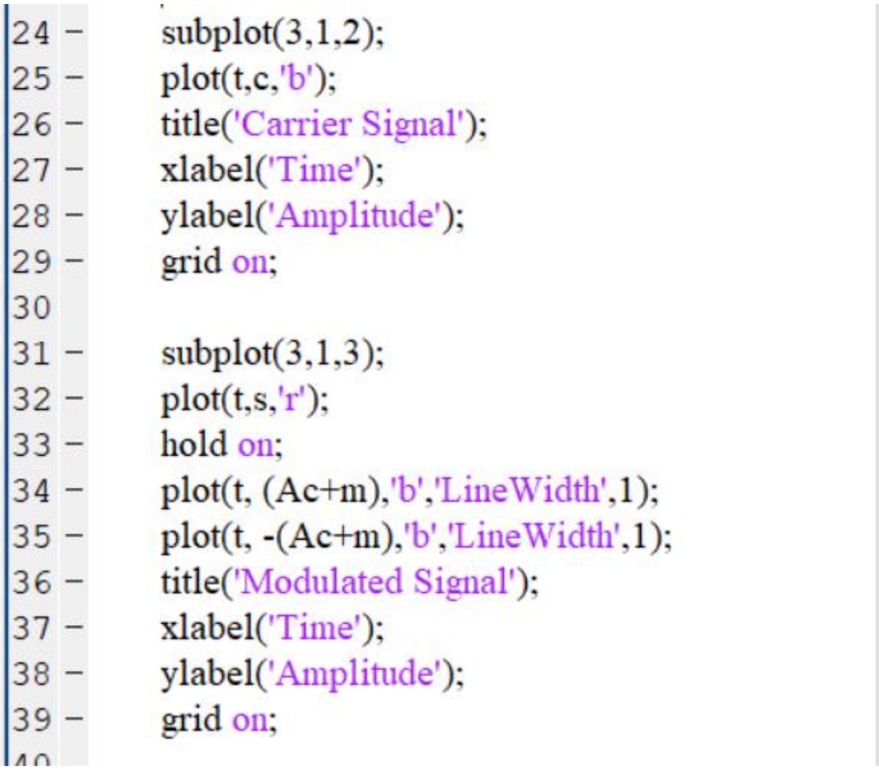
Problem 1: Implementation of Amplitude Modulation

Source Code:

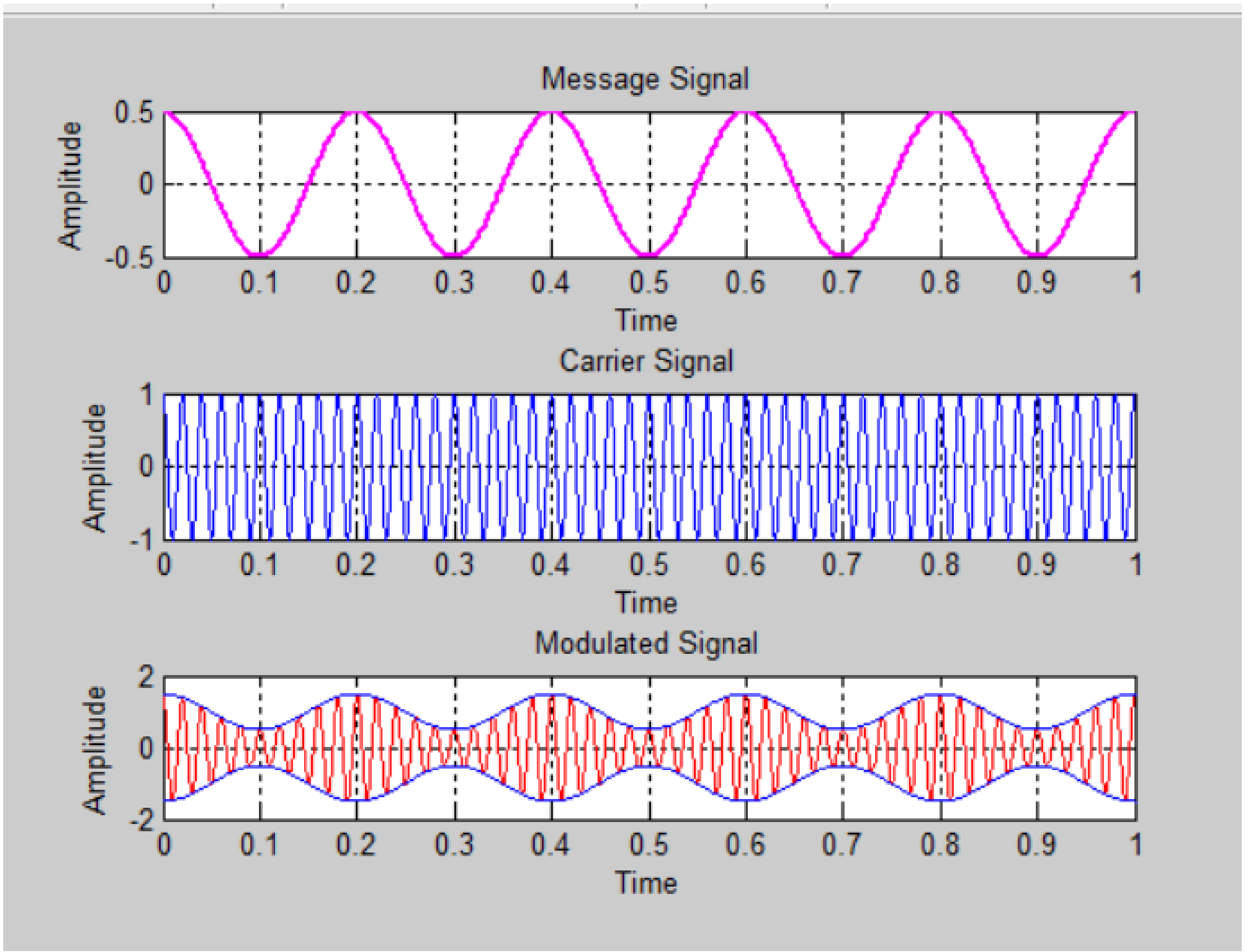

Editor - D:\Academic\2nd year\2nd semester\Data Communication Lab\Lab 5\amsir.m
ask.m
fsk.m
psk.m
amsir.m
+

```
clc;
clear all;
close all;

% Asin(2*pi*f*t*phase)
Am= 0.5; %amplitude of message signal
Ac= 1; %amplitude of carrier signal
fm=5; %Frequency of message signal
fc=50; %Frequency of carrier signal
t=0:0.0001:1; %Time Vector

m = Am* cos(2*pi*fm*t);
c= Ac* cos(2*pi*fc*t);
s = (Ac+m).* cos(2*pi*fc*t);

figure;
subplot(3,1,1);
plot(t,m,'m','LineWidth',1.5);
title('Message Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;
```



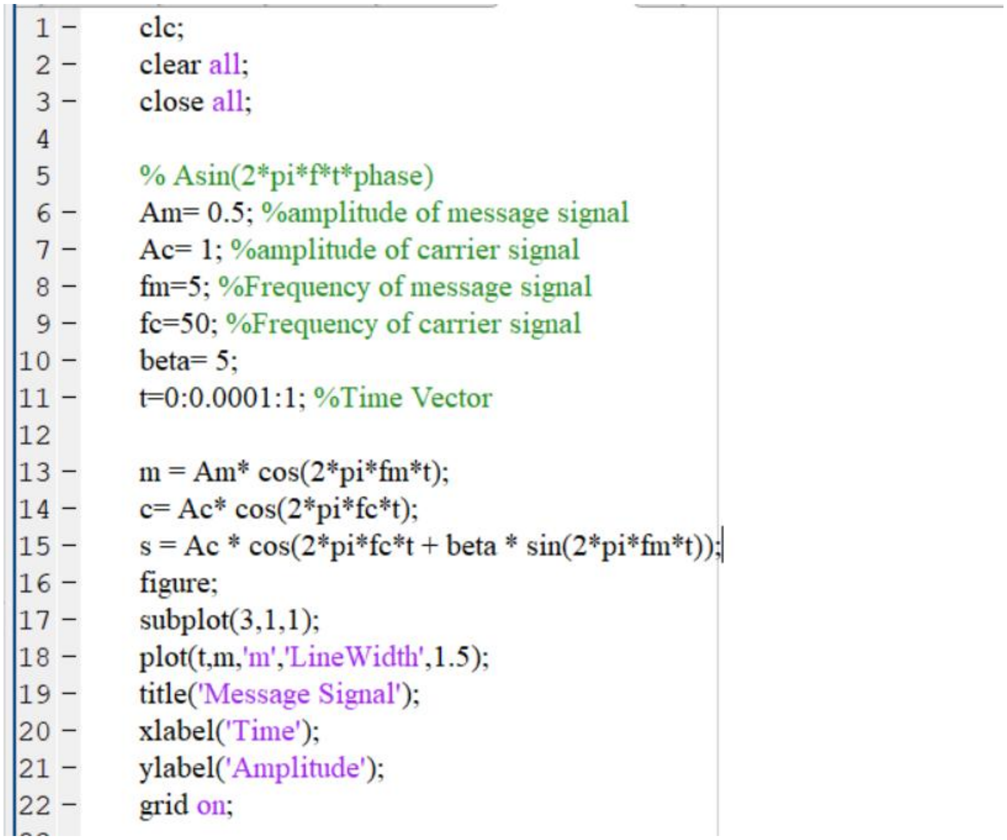
```
subplot(3,1,2);
plot(t,c,'b');
title('Carrier Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;

subplot(3,1,3);
plot(t,s,'r');
hold on;
plot(t, (Ac+m),'b','LineWidth',1);
plot(t, -(Ac+m),'b','LineWidth',1);
title('Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;
```

Output:

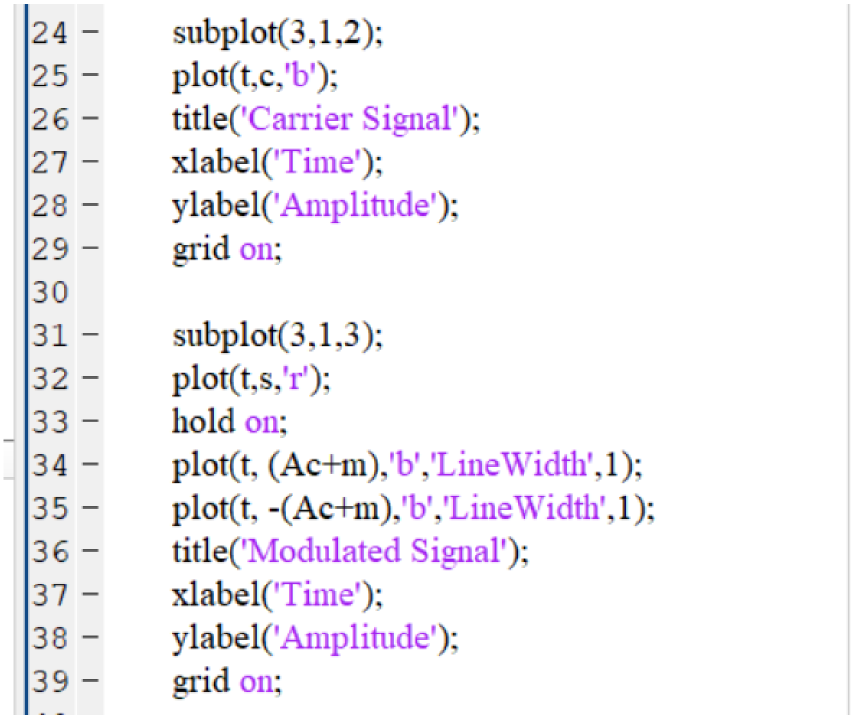

Message Signal
Amplitude
0.5
0
-0.5
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time
Carrier Signal
Amplitude
1
0
-1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time
Modulated Signal
Amplitude
2
0
-2
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time

Problem 2: Implementation of Frequency Modulation

Source Code:

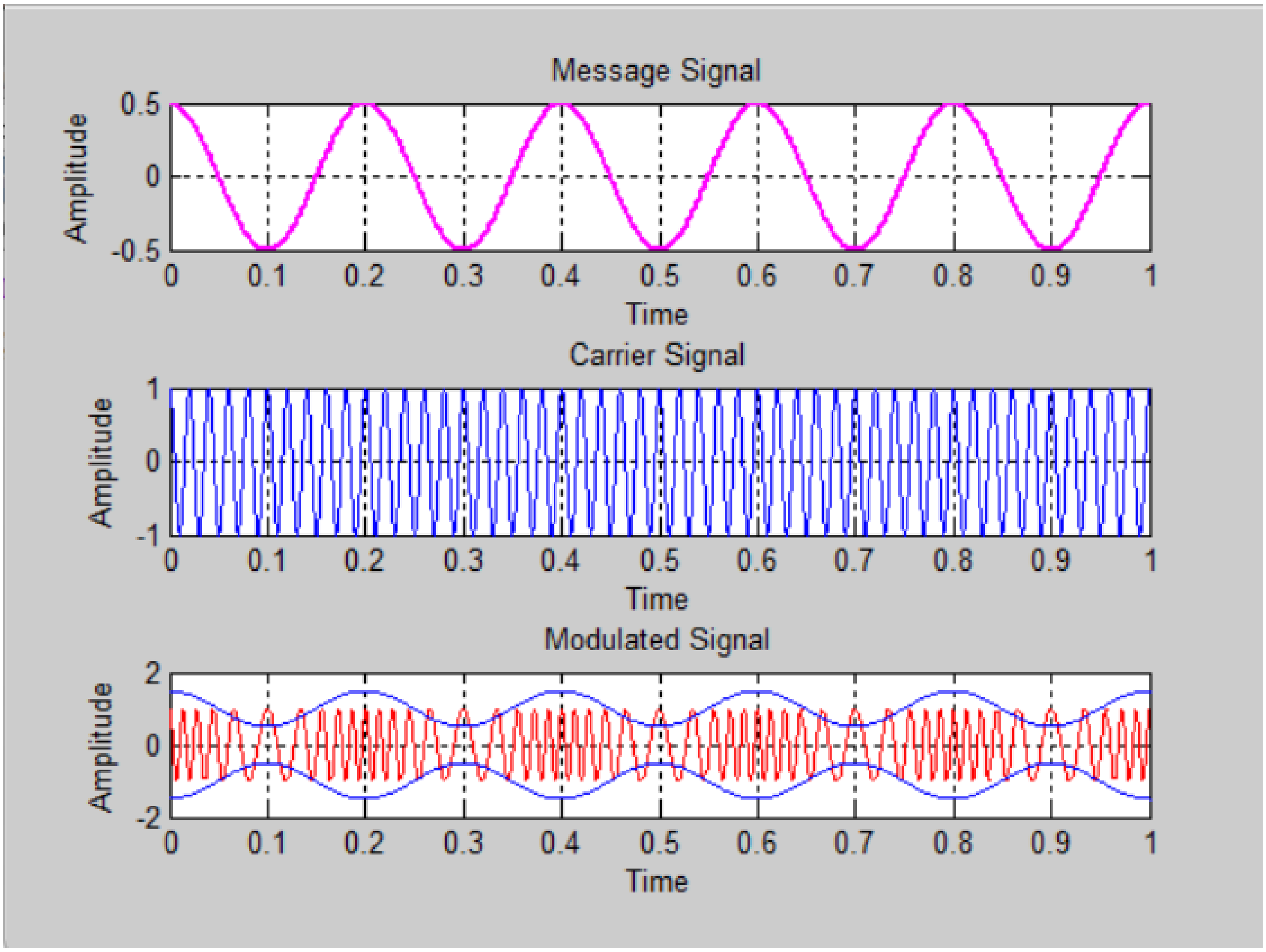


```
clc;
clear all;
close all;

% Asin(2*pi*f*t*phase)
Am= 0.5; %amplitude of message signal
Ac= 1; %amplitude of carrier signal
fm=5; %Frequency of message signal
fc=50; %Frequency of carrier signal
beta= 5;
t=0:0.0001:1; %Time Vector

m = Am* cos(2*pi*fm*t);
c= Ac* cos(2*pi*fc*t);
s = Ac * cos(2*pi*fc*t + beta * sin(2*pi*fm*t));
figure;
subplot(3,1,1);
plot(t,m,'m','LineWidth',1.5);
title('Message Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;
```



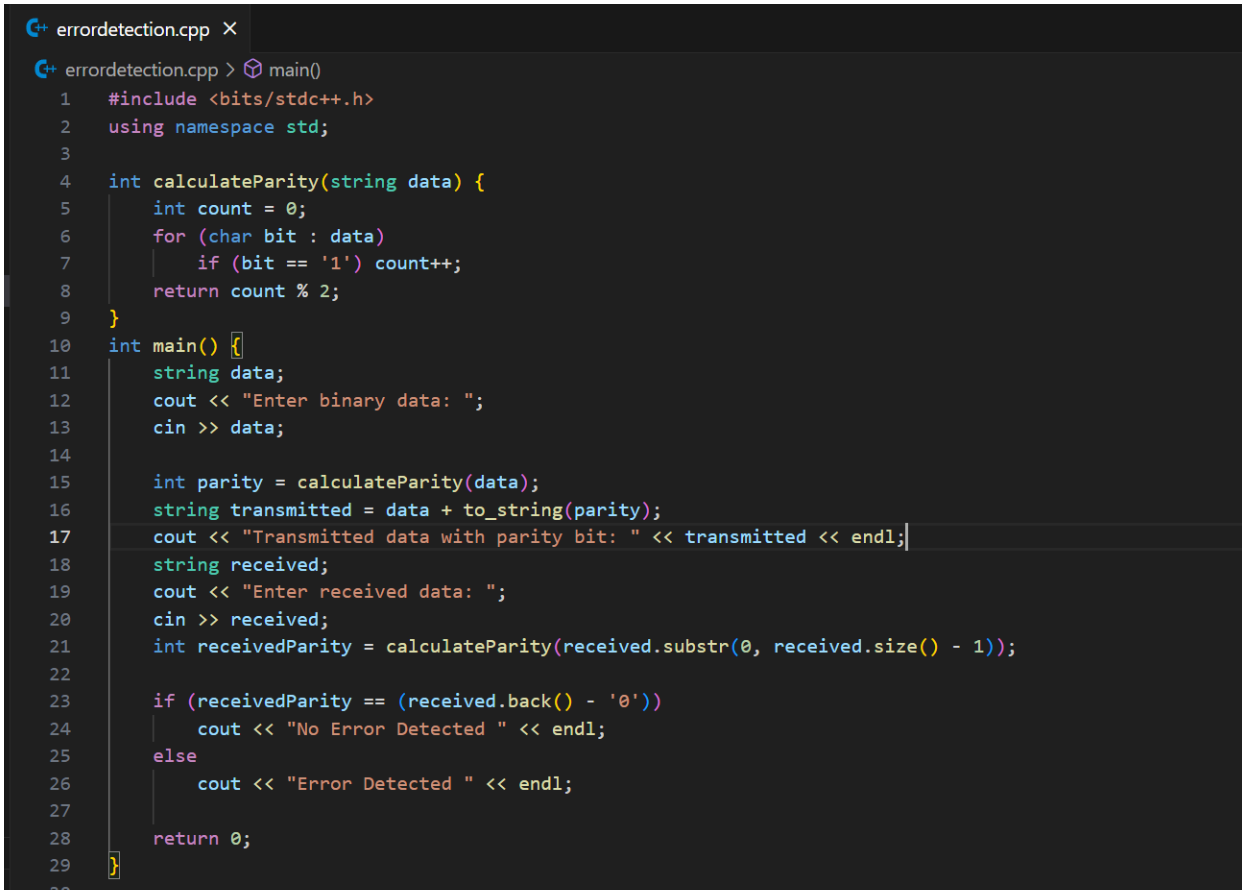
```
subplot(3,1,2);
plot(t,c,'b');
title('Carrier Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;

subplot(3,1,3);
plot(t,s,'r');
hold on;
plot(t, (Ac+m),'b','LineWidth',1);
plot(t, -(Ac+m),'b','LineWidth',1);
title('Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;
```

Output:


Message Signal
Amplitude
0.5
0
-0.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time
Carrier Signal
Amplitude
1
0
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time
Modulated Signal
Amplitude
2
0
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time

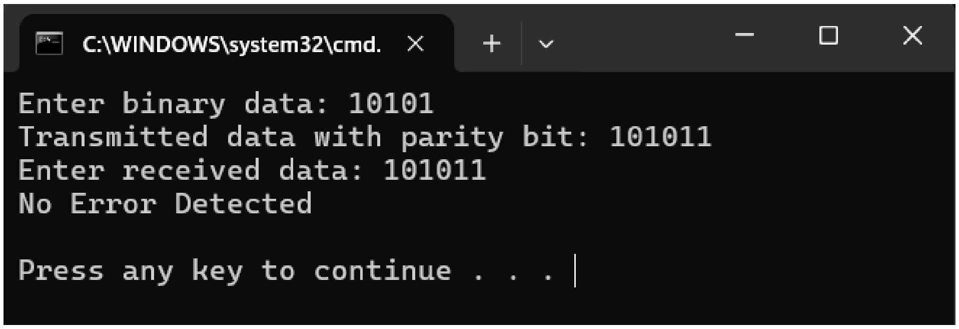
Problem 3: Implementation of Error Detection (parity bits)

Source Code:

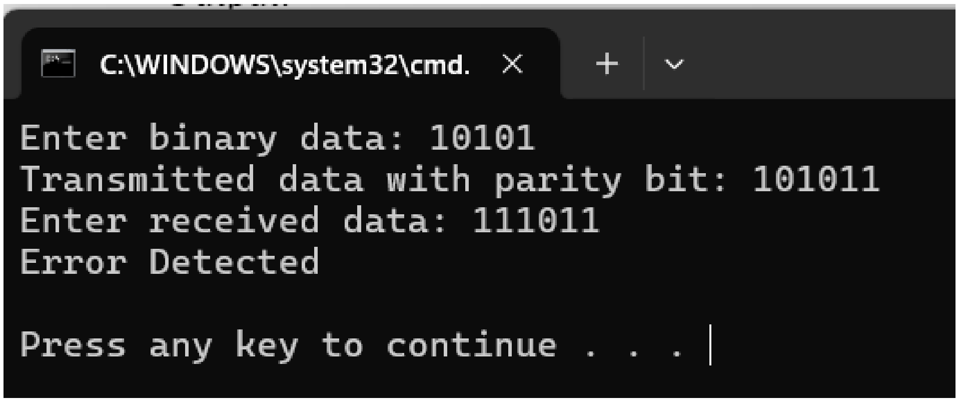


```cpp
#include <bits/stdc++.h>
using namespace std;

int calculateParity(string data) {
    int count = 0;
    for (char bit : data)
        if (bit == '1') count++;
    return count % 2;
}
int main() {
    string data;
    cout << "Enter binary data: ";
    cin >> data;

    int parity = calculateParity(data);
    string transmitted = data + to_string(parity);
    cout << "Transmitted data with parity bit: " << transmitted << endl;
    string received;
    cout << "Enter received data: ";
    cin >> received;
    int receivedParity = calculateParity(received.substr(0, received.size() - 1));

    if (receivedParity == (received.back() - '0'))
        cout << "No Error Detected " << endl;
    else
        cout << "Error Detected " << endl;

    return 0;
}
```

Output:



```
Enter binary data: 10101
Transmitted data with parity bit: 101011
Enter received data: 101011
No Error Detected

Press any key to continue . . .
```



```
Enter binary data: 10101
Transmitted data with parity bit: 101011
Enter received data: 111011
Error Detected

Press any key to continue . . .
```