

# Computer Networks- C & JAVA Programs

## C Programs

1)

The screenshot shows a C program in a code editor and its execution output in a console window. The code defines a sliding window protocol simulation with a window size of 5 and 45 frames to transmit. It prints the window size, the number of frames, and the sequence of frames sent. The output shows the frames being sent in a sliding window manner, with acknowledgements received by the sender. The process exits after 12.08 seconds.

```
#include<stdio.h>
int main()
{
    int w,i,f,frames[50];
    printf("Enter window size: ");
    scanf("%d",&w);
    printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);
    printf("\nEnter %d frames: ",f);
    for(i=1;i<=f;i++)
        scanf("%d",&frames[i]);
    printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n");
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",w);
    for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
    }
}
```

Output:

```
5
With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)
After sending 45 frames at each stage sender waits for acknowledgement sent by the receiver
1 2 3 4 5
Acknowledgement of above frames sent is received by sender
-----
Process exited after 12.08 seconds with return value 0
Press any key to continue . . .
```

2)

The screenshot shows a C program in a code editor and its execution output in a console window. The code takes 45 data bits as input and performs bit stuffing to ensure the resulting bit stream contains no consecutive 1s. The output shows the data bits, the stuffed bits, and the final bit stream. The process exits after 3.409 seconds.

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i=0,count=0;
    char databits[80];
    printf("Enter Data Bits: ");
    scanf("%s",databits);
    printf("\nData Bits After Bit stuffing: ");
    for(i=0; i<strlen(databits); i++)
    {
        if(databits[i]=='1')
            count++;
        else
        {
            count=0;
            printf("%c",databits[i]);
            if(count==5)
            {
                printf("0");
                count=0;
            }
        }
    }
    return 0;
}
```

Output:

```
Enter Data Bits: 45
Data Bits After Bit stuffing: 45
-----
Process exited after 3.409 seconds with return value 0
Press any key to continue . . .
```

# JAVA Programs

1)

The screenshot shows a web browser window with the URL `programiz.com/java-programming/online-compiler/`. The page features the Programiz logo and a navigation bar. The main content area is divided into two panels: a code editor on the left and an output window on the right. The code editor contains a Java program named `Main.java` that imports `java.time.LocalDateTime` and `java.time.format.DateTimeFormatter`, defines a `Main` class with a `main` method, and prints the current date and time in the format `dd-MM-yyyy HH:mm:ss`. The output window shows the result of running the program: `java -cp /tmp/YykKw6bGB Main` followed by `The current date and time is: 12-08-2023 07:41:25`. The bottom of the browser window shows a Windows taskbar with various icons and the system clock indicating 13:11 on 12-08-2023.

```
1- import java.time.LocalDateTime;
2 import java.time.format.DateTimeFormatter;
3
4- public class Main {
5+ public static void main(String[] args) {
6     LocalDateTime myDateObj = LocalDateTime.now();
7     DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern
8         ("dd-MM-yyyy HH:mm:ss");
9
10    String formattedDate = myDateObj.format(myFormatObj);
11    System.out.println("The current date and time is: " +
12        formattedDate);
13 }
```

Output

```
java -cp /tmp/YykKw6bGB Main
The current date and time is: 12-08-2023 07:41:25
```

2)

The screenshot shows a C++ IDE window with the file `CN-1Q.cpp` open. The code implements a CRC error detection algorithm. It defines a generator polynomial and a data string, pads the data with zeros, calculates the CRC value, and then checks the received data against the calculated CRC. The output window shows the results of running the program: `Enter data to be transmitted: 45`, `Enter the Generating polynomial: 7`, `Data padded with n-1 zeros : 45`, `CRC or Check value is :`, `Final data to be sent : 45`, `Enter the received data: 23`, `Data received: 23`, `No error detected`, and `Process exited after 18.57 seconds with return value 0`. The bottom of the IDE window shows a Windows taskbar with various icons and the system clock indicating 12:57 on 12-08-2023.

```
1 // Include headers
2 #include<stdio.h>
3 #include<string.h>
4 // length of the generator polynomial
5 #define N strlen(gen_poly)
6 // data to be transmitted and received
7 char data[28];
8 // CRC value
9 char check_value[28];
10 // generator polynomial
11 char gen_poly[10];
12 // variables
13 int data_length,i,j;
14 // function that performs XOR operation
15 void XOR(){
16     // if both bits are the same, the output is 0
17     // if the bits are different the output is 1
18     for(j = 1;j < N; j++){
19         check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');
20     }
21 }
22 // Function to check for errors on the receiver side
23 void receiver(){
24     // get the received data
25     printf("Enter the received data: ");
26     scanf("%s", data);
```

Enter data to be transmitted: 45

Enter the Generating polynomial: 7

-----

Data padded with n-1 zeros : 45

-----

CRC or Check value is :

-----

Final data to be sent : 45

-----

Enter the received data: 23

-----

Data received: 23

No error detected

-----

Process exited after 18.57 seconds with return value 0

Press any key to continue . . .