Name : Bijay Regmi
Regd. No : 210913032

Q1. Apply simple FEC on a given number of data chunks. The transmitter program must accept the data chunks from the user and add the redundant chunk. The original and redundant chunks must be displayed. The receiver program must receive the data chunks with/without error, apply error correction if required and display the original data chunks.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n, b, n1;
    int i, j;
    int a[20][20], r[20][20];

    printf("enter no of data chunks to be sent:\n");
    scanf("%d", &n);
    printf("enter no of bits to be sent in each data chunk:\n");
    scanf("%d", &b);
    printf("\nAT SENDER:\n");
    printf("enter data chunks to be sent (in binary) :\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < b; j++)
            scanf("%d", &a[i][j]);
    printf("adding redundant data chunk\n");
    printf("redundant bits added\n");
    printf("transmitting data chunks:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < b; j++)
            printf("%d ", a[i][j]);
        printf("\t");
    }
```

```
printf("\n");
printf("transmitting redundant bits:\n");
for (i = 0; i < n - 1; i++)
    for (j = 0; j < b; j++)
        if (a[i][j] == a[i + 1][j])
            a[i + 1][j] = 0;
        else
            a[i + 1][j] = 1;
for (i = 0; i < n; i++)
    for (j = 0; j < b; j++)
        if (i == n - 1)
            printf("%d ", a[i][j]);
printf("\t");
printf("\n\nAT RECEIVER:\n");
printf("received data is:\n");
printf("Enter the number of chunks of data received:\n");
scanf("%d", &n1);
if (n1 != n - 1)
{
    printf("lost packets cannot be reconstructed\n");
    exit(0);
}

for (i = 0; i < n1; i++)
{
    printf("enter bits of chunk %d:\n", i + 1);
    for (j = 0; j < b; j++)
        scanf("%d", &a[i][j]);
}
printf("enter bits of redundant chunk recieved:\n");
for (j = 0; j < b; j++)
    printf("%d ", a[n - 1][j]);
printf("\napplying error correction\n");
printf("recieved data after error correction:\n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < b; j++)
    {
        if (i < n - 1)
            printf("%d ", a[i][j]);
    }
    printf("\t");
}
for (i = 0; i < n; i++)
```

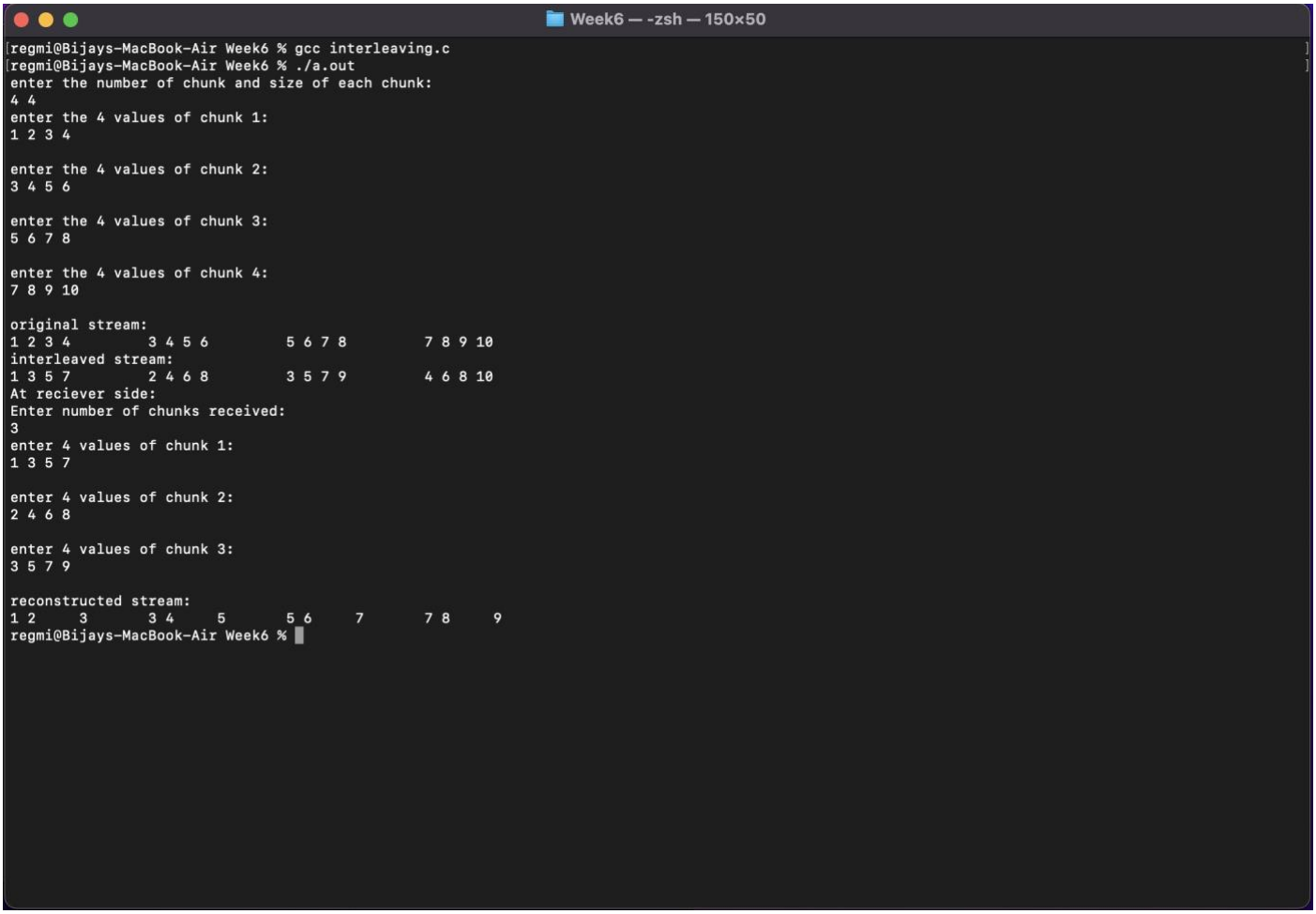
```
  {
    for (j = 0; j < b; j++)
    {
      r[i][j] = a[i][j];
      a[i + 1][j] = a[i + 1][j] ^ a[i][j];
      if (i == n - 1)
        printf("%d ", r[i][j]);
    }
  }
  return 0;
}
```

```
Week6 — -zsh — 150×50

regmi@Bijays-MacBook-Air Week6 % gcc FEC.c
regmi@Bijays-MacBook-Air Week6 % ./a.out
enter no of data chunks to be sent:
4
enter no of bits to be sent in each data chunk:
4

AT SENDER:
enter data chunks to be sent (in binary) :
1 0 1 0
1 1 0 0
1 1 1 0
1 1 1 1
adding redundant data chunk
redundant bits added
transmitting data chunks:
1 0 1 0        1 1 0 0        1 1 1 0        1 1 1 1
transmitting redundant bits:
0 1 1 1

AT RECEIVER:
received data is:
Enter the number of chunks of data received:
3
enter bits of chunk 1:
1 0 1 0
enter bits of chunk 2:
1 1 0 0
enter bits of chunk 3:
1 1 1 0
enter bits of redundant chunk recieved:
0 1 1 1
applying error correction
recieved data after error correction:
1 0 1 0        1 1 0 0        1 1 1 0              1 1 1 1 %
regmi@Bijays-MacBook-Air Week6 %
```

Q2 Apply interleaving mechanism on a given number of data chunks. The transmitter program must accept the original stream and display the interleaved stream. The receiver program must accept the received stream with/without loss and display the reconstructed stream.

```c
#include <stdio.h>
int main()
{
    int n, b, size, n1;
    int i, j;
    int a[20];
    printf("enter the number of chunk and size of each chunk: \n");
    scanf("%d%d", &n, &b);
    size = n * b;
    for (i = 0; i < n; i++)
    {
        printf("enter the %d values of chunk %d:\n", b, i + 1);
        for (j = (i * b); j < (i * b) + b; j++)
        {
            scanf("%d", &a[j]);
        }
        printf("\n");
    }
    printf("original stream:\n");
    for (i = 0; i < size; i++)
    {
        if ((i + 1) % b == 0)
            printf("%d\t\t", a[i]);
        else
            printf("%d ", a[i]);
```

```
    }
    printf("\n");

    printf("interleaved stream:\n");
    for (i = 0; i < b; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%d ", a[i + (j * b)]);
        }
        printf("\t");
    }
    printf("\n");
    printf("At reciever side:\n");
    printf("Enter number of chunks received:\n");
    scanf("%d", &n1);
    for (i = 0; i < n1; i++)
    {
        printf("enter %d values of chunk %d:\n", b, i + 1);
        for (j = (i * n); j < (i * n) + n; j++)
        {
            scanf("%d", &a[j]);
        }
        printf("\n");
    }
    printf("reconstructed stream:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n1; j++)
        {
            if (j == 2)
                printf("\t");
            printf("%d ", a[i + (j * n)]);
        }
        printf("\t");
    }
    printf("\n");
    return 0;
}
```


Week6 — -zsh — 150×50
regmi@Bijays-MacBook-Air Week6 % gcc interleaving.c
regmi@Bijays-MacBook-Air Week6 % ./a.out
enter the number of chunk and size of each chunk:
4 4
enter the 4 values of chunk 1:
1 2 3 4
enter the 4 values of chunk 2:
3 4 5 6
enter the 4 values of chunk 3:
5 6 7 8
enter the 4 values of chunk 4:
7 8 9 10
original stream:
1 2 3 4 3 4 5 6 5 6 7 8 7 8 9 10
interleaved stream:
1 3 5 7 2 4 6 8 3 5 7 9 4 6 8 10
At reciever side:
Enter number of chunks received:
3
enter 4 values of chunk 1:
1 3 5 7
enter 4 values of chunk 2:
2 4 6 8
enter 4 values of chunk 3:
3 5 7 9
reconstructed stream:
1 2 3 3 4 5 5 6 7 7 8 9
regmi@Bijays-MacBook-Air Week6 %