Shudhanshu Shrotriya

500086221

Batch-4

**Data Communication & Computer Networks Lab**

**Experiment- 2**

**Write a program for Bit stuffing and De-stuffing in c language.**

**Bit Stuffing:**

In data link frame, the flag sequencing for the most part contains at least six continuous 1s. To separate the message from the flag if there should be an occurrence of a similar sequence, a single bit is added or stuffed in the message. At the point when a 0 cycle is trailed by five successive 1bits in the message, an additional 0 bit is added or stuffed toward the ending of the five continuous 1s.

**Example:**

Input:

n = 6,

arr\_1[] = {1, 1, 1, 1, 1, 0}

Output:

1111100

**Explanation:**

When the traversal of array was occurring, 5 consecutive 1’s were encountered after the 4th index of the given array. Hence, a 0 bit has been inserted into the stuffed array after the 4th index.

Another Example:

Input:

n = 6,

arr\_1[] = {1, 0, 0, 0, 1, 0}

Output:

100010

**Explanation:**

Here during the traversal of the array, we do not encounter 5 consecutive 1’s, hence no 0 bit is stuffed into the array.

**Approach:**

The main aim here is to verify if the given array consists of 5 consecutive 1’s. And for that we use the following steps:

Firstly, we initialize the array arr\_2[] which stores the stuffed array. Also, then we create a variable cnt which stores the count of the consecutive 1’s.

Then, we traverse in a while loop using a variable i in the range [0, n) and perform the below tasks:

If arr\_1[i] is 1 then check for the next 4 bits if they are set bits as well.

If they are, then insert a 0 bit after inserting all the 5 set bits into the array arr\_2[], Else, insert the value of arr\_1[i] into the array arr\_2[].

**Code for Bit Stuffing:**

#include <stdio.h>

#include <string.h>

#define MAX\_SIZE 100

// here I am creating the function for bit-stuffing

void bit\_stuff(int n, int arr\_1[])

{

int arr\_2[MAX\_SIZE];

int i = 0, j = 0, k, cnt = 1;

while (i < n) {

if (arr\_1[i] == 1) {

arr\_2[j] = arr\_1[i];

for (k = i + 1;

arr\_1[k] == 1 && k < n && cnt < 5; k++) {

j++;

arr\_2[j] = arr\_1[k];

cnt++;

if (cnt == 5) {

j++;

arr\_2[j] = 0;

}

i = k;

}

}

else {

arr\_2[j] = arr\_1[i];

}

i++;

j++;

}

for (i = 0; i < j; i++)

printf("%d", arr\_2[i]);

}

int main()

{

// here I am requesting the user for array input on which the function bit\_stuff will work

int n;

printf("Please enter number of array elements:\t");

scanf("%d", &n);

int arr\_1[MAX\_SIZE];

printf("Enter %d elements in the array : \n", n);

for(int i=0; i<n; i++)

{

scanf("%d", &arr\_1[i]);

}

// here I am displaying the user his entered array so that he can refer to the array after bit stuffing

printf("\nThe array elements entered by you are: \n");

for(int i=0; i<n; i++)

{

printf("%d", arr\_1[i]);

}

// here we call the bit\_stuffing function and also print the array after the bit stuffing operation

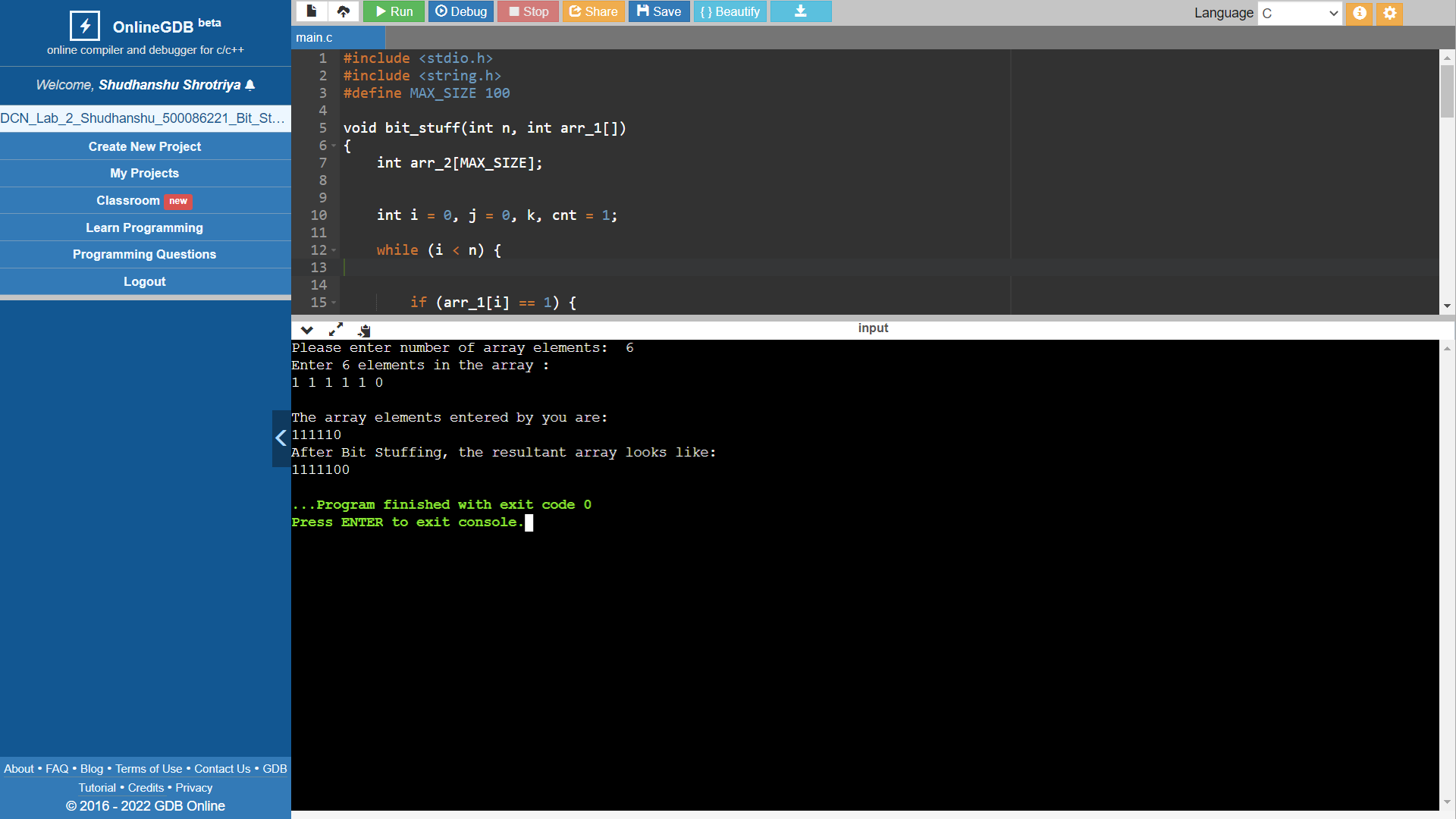
printf("\nAfter Bit Stuffing, the resultant array looks like: \n");

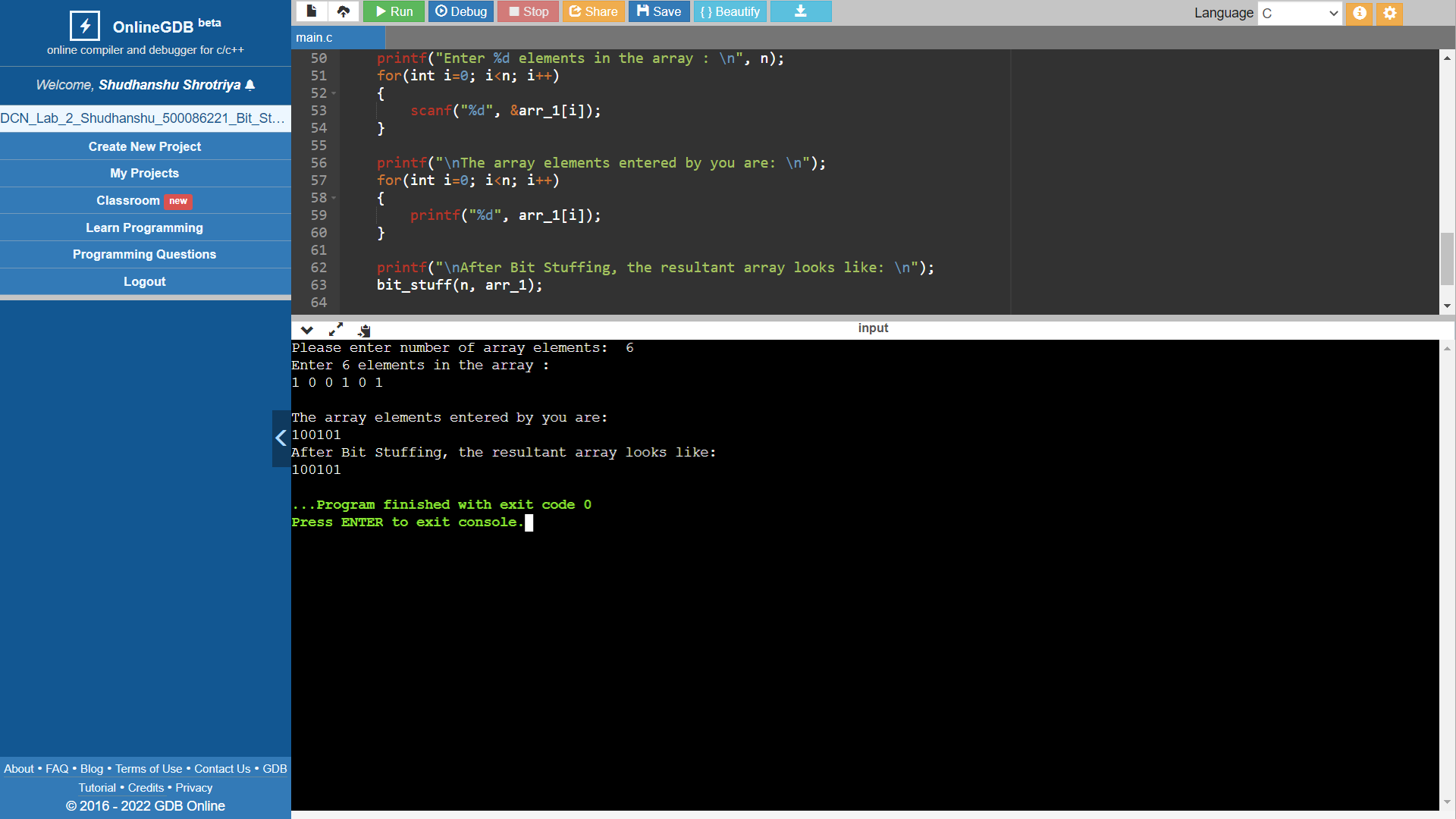
bit\_stuff(n, arr\_1);

return 0;

}

**Output:**





**Bit De-Stuffing:**

Bit De-Stuffing is a reverse process of Bit Stuffing. It is the process of undoing the progressions in the array made during the bit stuffing process i.e, eliminating the additional 0 bit after coming across 5 consecutive 1's.

**Example:**

Input:

n = 7, arr\_1[] = {1, 1, 1, 1, 1, 0, 0}

Output:

111110

**Explanation:**

When we traverse the array, then we come across 5 consecutive 1’s after the 4th index of the given array. Hence, the next 0 bit must be removed to de-stuff the array.

**Another Example:**

Input:

n = 6, arr\_1[] = {1, 0, 0, 0, 1, 0}

Output:

100010

**Approach:**

The main aim here is to verify if the given array consists of 5 consecutive 1’s. And for that we use the following steps:

Firstly, we initialize the array arr\_2[] which stores the stuffed array. Also, then we create a variable cnt which stores the count of the consecutive 1’s.

Then, we traverse in a while loop using a variable i in the range [0, n) and perform the below tasks:

If arr\_1[i] is 1 then check for the next 4 bits if they are set bits as well.

If they are, then we skip the next bit in the array.

**Code for bit De-Stuffing:**

#include <stdio.h>

#include <string.h>

#define MAX\_SIZE 100

// here I am writing the function for bit de-stuffing

void bit\_destuff(int n, int arr\_1[])

{

int arr\_2[MAX\_SIZE];

int i = 0, j = 0, k, cnt = 1;

while (i < n) {

if (arr\_1[i] == 1) {

arr\_2[j] = arr\_1[i];

for (k = i + 1;

arr\_1[k] == 1 && k < n && cnt < 5; k++) {

j++;

arr\_2[j] = arr\_1[k];

cnt++;

if (cnt == 5) {

k++;

}

i = k;

}

}

else {

arr\_2[j] = arr\_1[i];

}

i++;

j++;

}

for (i = 0; i < j; i++)

printf("%d", arr\_2[i]);

}

int main()

{

//here I am requesting the user for array input on which the bit de-stuffing operation is to be done

int n;

printf("Please enter number of array elements:\t");

scanf("%d", &n);

int arr\_1[MAX\_SIZE];

printf("Enter %d elements in the array : \n", n);

for(int i=0; i<n; i++)

{

scanf("%d", &arr\_1[i]);

}

// here I am displaying the user his entered array so that he can refer to the array after bit de-stuffing

printf("\nThe array elements entered by you are: \n");

for(int i=0; i<n; i++)

{

printf("%d", arr\_1[i]);

}

// here I am calling the bit\_destuff function to perform the operation of bit de-stuffing

printf("\nAfter Bit De-Stuffing, the resultant array looks like: \n");

bit\_destuff(n, arr\_1);

return 0;

}

**Output:**

