1. ***Write a program for Hamming code generation.***

***CODE:***

**#include <bits/stdc++.h>**

**using namespace std;**

**vector<int> generateHammingCode(**

**vector<int> msgBits, int m, int r)**

**{**

**vector<int> hammingCode(r + m);**

**for (int i = 0; i < r; ++i) {**

**hammingCode[pow(2, i) - 1] = -1;**

**}**

**int j = 0;**

**for (int i = 0; i < (r + m); i++) {**

**if (hammingCode[i] != -1) {**

**hammingCode[i] = msgBits[j];**

**j++;**

**}**

**}**

**for (int i = 0; i < (r + m); i++) {**

**if (hammingCode[i] != -1)**

**continue;**

**int x = log2(i + 1);**

**int one\_count = 0;**

**for (int j = i + 2;**

**j <= (r + m); ++j) {**

**if (j & (1 << x)) {**

**if (hammingCode[j - 1] == 1) {**

**one\_count++;**

**}**

**}**

**}**

**if (one\_count % 2 == 0) {**

**hammingCode[i] = 0;**

**}**

**else {**

**hammingCode[i] = 1;**

**}**

**}**

**return hammingCode;**

**}**

**void findHammingCode(vector<int>& msgBit)**

**{**

**int m = msgBit.size();**

**int r = 1;**

**while (pow(2, r) < (m + r + 1)) {**

**r++;**

**}**

**vector<int> ans**

**= generateHammingCode(msgBit, m, r);**

**cout << "Message bits are: ";**

**for (int i = 0; i < msgBit.size(); i++)**

**cout << msgBit[i] << " ";**

**cout << "\nHamming code is: ";**

**for (int i = 0; i < ans.size(); i++)**

**cout << ans[i] << " ";**

**}**

**int main()**

**{**

**vector<int> msgBit = { 0, 1, 0, 1 };**

**findHammingCode(msgBit);**

**return 0;**

**}**

***OUTPUT:***

**Message bits are: 0 1 0 1**

**Hamming code is: 0 1 0 0 1 0 1**

***5. Write a program for congestion control using the leaky bucket algorithm***

***CODE:***

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int no\_of\_queries, storage, output\_pkt\_size;**

**int input\_pkt\_size, bucket\_size, size\_left;**

**storage = 0;**

**no\_of\_queries = 4;**

**bucket\_size = 10;**

**input\_pkt\_size = 4;**

**output\_pkt\_size = 1;**

**for (int i = 0; i < no\_of\_queries; i++)**

**{**

**size\_left = bucket\_size - storage;**

**if (input\_pkt\_size <= size\_left) {**

**storage += input\_pkt\_size;**

**}**

**else {**

**printf("Packet loss = %d\n", input\_pkt\_size);**

**}**

**printf("Buffer size= %d out of bucket size= %d\n",**

**storage, bucket\_size);**

**storage -= output\_pkt\_size;**

**}**

**return 0;**

**}**

**OUTPUT:**

**Buffer size= 4 out of bucket size= 10**

**Buffer size= 7 out of bucket size= 10**

**Buffer size= 10 out of bucket size= 10**

**Packet loss = 4**

**Buffer size= 9 out of bucket size= 10**