**COMPUTER NETWORKS-LAB EXPERIMENTS(1-7)**

1)Implement the data link layer framing methods such as character, character-stuffing and bit stuffing

/\*QUESTION-1(A): Character count\*/

#include<stdio.h>

#include<string.h>

int main(){

int n,i,j,c=0,count=0;

char str[100];

printf("Enter the string: ");

scanf("%s", str);

printf("Enter the number of frames:");

scanf("%d",&n);

int frames[n];

printf("Enter the frame size of the frames:\n");

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

printf("Frame %d:",i);

scanf("%d",&frames[i]);

}

printf("\nThe number of frames:%d\n",n);

c = 0;

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

printf("The content of the frame %d:",i);

j=0;

count = 0;

while(c < strlen(str) && j < frames[i]){

printf("%c",str[c]);

if(str[c]!='\0'){

count++;

}

c=c+1;

j=j+1;

}

printf("\nSize of frame %d: %d\n\n",i,count);

}

return 0;

}

/\*QUESTION-1(B): Character stuffing\*/

#include <stdio.h>

#include <string.h>

#define FRAME\_START 0x7E // Start of Frame

#define FRAME\_END 0x7E // End of Frame

#define STUFFING\_CHAR 0x7D // Escape character

#define ESCAPED\_CHAR 0x20 // Value to append after STUFFING\_CHAR

void stuffCharacters(char \*input, char \*output) {

int i, j = 0;

int length = strlen(input);

// Adding the frame start character

output[j++] = FRAME\_START;

for (i = 0; i < length; i++) {

if (input[i] == FRAME\_START || input[i] == FRAME\_END || input[i] == STUFFING\_CHAR) {

output[j++] = STUFFING\_CHAR; // Stuffing character

output[j++] = input[i] ^ ESCAPED\_CHAR; // Escape the character

} else {

output[j++] = input[i]; // Normal character

}

}

// Adding the frame end charactera

output[j++] = FRAME\_END;

output[j] = '\0'; // Null-terminate the output string

}

void unstuffCharacters(char \*input, char \*output) {

int i, j = 0;

int length = strlen(input);

// Skip frame start

i = 1;

while (i < length - 1) { // Skip frame end

if (input[i] == STUFFING\_CHAR) {

// Unstuffing character

output[j++] = input[i + 1] ^ ESCAPED\_CHAR;

i += 2; // Move past the stuffed character

} else {

output[j++] = input[i++];

}

}

output[j] = '\0'; // Null-terminate the output string

}

int main() {

char input[256]; // Buffer for input message

char stuffed[512]; // Buffer for stuffed message (larger to accommodate potential expansion)

char unstuffed[256]; // Buffer for unstuffed message

printf("Enter your message: ");

fgets(input, sizeof(input), stdin); // Read input from the keyboard

// Remove newline character if present

input[strcspn(input, "\n")] = 0;

printf("Original Message: %s\n", input);

// Perform character stuffing

stuffCharacters(input, stuffed);

printf("Stuffed Message: ");

for (int i = 0; stuffed[i] != '\0'; i++) {

printf("%02X ", (unsigned char)stuffed[i]);

}

printf("\n");

// Perform character unstuffing

unstuffCharacters(stuffed, unstuffed);

printf("Unstuffed Message: %s\n", unstuffed);

return 0;

}

/\*QUESTION-1(C): Character bit stuffing\*/

#include<stdio.h>

#include<string.h>

void bitStuffing(char \*str) {

int i, count = 0;

char stuffedStr[100] = ""; // initialize an empty string to store the stuffed bits

for (i = 0; i < strlen(str); i++) {

if (str[i] == '1') {

count++;

if (count == 5) {

strcat(stuffedStr, "0"); // stuff a '0' bit

count = 0;

}

} else {

count = 0;

}

char temp[2];

sprintf(temp, "%c", str[i]);

strcat(stuffedStr, temp);

}

printf("Original string: %s\n", str);

printf("Bit-stuffed string: %s\n", stuffedStr);

}

int main() {

char str[100];

printf("Enter a binary string: ");

scanf("%s", str);

bitStuffing(str);

return 0;

}

2)Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRCCCIP 3

/\*QUESTION-2: CRC Code\*/

#include <stdio.h>

#include <string.h>

#define MAX\_LEN 28

char t[MAX\_LEN], cs[MAX\_LEN], g[MAX\_LEN];

int a, e, c, b;

void xor() {

for (c = 1; c < strlen(g); c++)

cs[c] = ((cs[c] == g[c]) ? '0' : '1');

}

void crc() {

for (e = 0; e < strlen(g); e++)

cs[e] = t[e];

do {

if (cs[0] == '1') {

xor();

}

for (c = 0; c < strlen(g) - 1; c++)

cs[c] = cs[c + 1];

cs[c] = t[e++];

} while (e <= a + strlen(g) - 1);

}

int main() {

int flag = 0;

do {

printf("\n1. CRC-12\n2. CRC-16\n3. CRC-CCITT\n4. Exit\n\nEnter your option: ");

scanf("%d", &b);

switch (b) {

case 1:

strcpy(g, "1100000001111");

break;

case 2:

strcpy(g, "11000000000000101");

break;

case 3:

strcpy(g, "10001000000100001");

break;

case 4:

return 0;

default:

printf("Invalid option. Please try again.\n");

continue;

}

printf("\nEnter data: ");

scanf("%s", t);

printf("\n-----------------------\n");

printf("Generating polynomial: %s\n", g);

a = strlen(t);

// Append zeros to the data

for (e = a; e < a + strlen(g) - 1; e++)

t[e] = '0';

t[e] = '\0'; // Null terminate the string

printf("--------------------------\n");

printf("Modified data is: %s\n", t);

printf("-----------------------\n");

crc();

printf("Checksum is: %s\n", cs);

// Prepare the final codeword

for (e = a; e < a + strlen(g) - 1; e++)

t[e] = cs[e - a];

printf("-----------------------\n");

printf("Final codeword is: %s\n", t);

printf("------------------------\n");

// Error detection option

printf("Test error detection (0: yes, 1: no)?: ");

scanf("%d", &e);

if (e == 0) {

do {

printf("\n\tEnter the position where error is to be inserted: ");

scanf("%d", &e);

} while (e <= 0 || e > a + strlen(g) - 1);

t[e - 1] = (t[e - 1] == '0') ? '1' : '0'; // Toggle the bit

printf("-----------------------\n");

printf("\nErroneous data: %s\n", t);

}

crc();

for (e = 0; (e < strlen(g) - 1) && (cs[e] != '1'); e++);

if (e < strlen(g) - 1)

printf("Error detected\n\n");

else

printf("No error detected\n\n");

printf("-----------------------\n");

} while (flag != 1);

return 0;

}

3)Develop a simple data link layer that performs flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism

/\*QUESTION-3: Sliding Window protocol\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdbool.h>

#define MAX\_SEQ 4 // Maximum sequence number (0 to 7)

#define WINDOW\_SIZE 4 // Size of the sliding window

// Packet structure

typedef struct {

int seq\_num; // Sequence number

bool ack; // Acknowledgment flag

} Packet;

// Function prototypes

void sender();

void receiver();

void send\_packet(Packet packet);

bool receive\_packet(Packet packet);

bool simulate\_packet\_loss();

int main() {

// Start sender and receiver (in a real implementation, these would run in separate threads or processes)

sender();

receiver();

return 0;

}

void sender() {

int next\_seq\_num = 0; // Next sequence number to send

int acked = 0; // Last acknowledged packet

int window\_start = 0; // Start of the sliding window

while (window\_start <= MAX\_SEQ) {

// Send packets within the window

while (next\_seq\_num < window\_start + WINDOW\_SIZE && next\_seq\_num <= MAX\_SEQ) {

Packet packet = {next\_seq\_num, false};

send\_packet(packet);

next\_seq\_num++;

sleep(1); // Simulate time taken to send a packet

}

// Simulate receiving an acknowledgment (in real systems this would be from the receiver)

for (int i = acked; i < next\_seq\_num; i++) {

if (receive\_packet((Packet){i, true})) {

acked++;

}

}

// Slide the window

window\_start = acked;

next\_seq\_num = window\_start; // Move next\_seq\_num to the next unacknowledged packet

}

printf("Sender finished transmitting all packets.\n");

}

void receiver() {

for (int i = 0; i <= MAX\_SEQ; i++) {

sleep(1); // Simulate processing time for receiving a packet

if (simulate\_packet\_loss()) {

printf("Receiver: Lost packet with seq\_num %d\n", i);

continue; // Simulate loss

}

printf("Receiver: Received packet with seq\_num %d\n", i);

}

}

void send\_packet(Packet packet) {

printf("Sender: Sending packet with seq\_num %d\n", packet.seq\_num);

}

bool receive\_packet(Packet packet) {

// Simulate acknowledgment logic

if (packet.ack) {

printf("Receiver: Acknowledgment for packet with seq\_num %d\n", packet.seq\_num);

return true;

}

return false;

}

bool simulate\_packet\_loss() {

// Randomly simulate packet loss (30% chance)

return (rand() % 10) < 3;

}

4)Implement Dijkstra’s algorithm to compute the shortest path through a network

/\*QUESTION-4: Dijkstra's Algorithm\*/

#include<stdio.h>

#include<stdbool.h>

#include<limits.h>

#define V 9

int minDistance(int dist[] ,bool sptset[]){

int min = INT\_MAX;

int min\_index;

for(int v=0;v<V;v++){

if(sptset[v] == false && dist[v]<=min){

min = dist[v];

min\_index = v;

}

}

return min\_index;

}

void printSolution(int dist[]){

printf("Vertex\t\tDistance from source\n");

for(int i=0;i<V;i++){

printf("%d\t\t\t\t %d\n",i,dist[i]);

}

}

void dijkstra(int graph[V][V],int src){

int dist[V];

bool sptset[V];

for(int i=0;i<V;i++){

dist[i] = INT\_MAX;

sptset[1]=false;

}

dist[src]=0;

for(int count=0;count<V-1;count++){

int u = minDistance(dist,sptset);

sptset[u]=true;

for(int v=0;v<V;v++){

if(!sptset[v] && graph[u][v] && dist[u]!=INT\_MAX && dist[u]+graph[u][v] < dist[v])

dist[v] = dist[u]+graph[u][v];

}

}

printSolution(dist);

}

int main(){

int graph[V][V] = {{0,4,0,0,0,0,0,8,0},

{4,0,8,0,0,0,0,11,0},

{0,8,0,7,0,4,0,0,2},

{0,0,7,0,9,14,0,0,0},

{0,0,0,9,0,10,0,0,0},

{0,0,4,14,10,0,2,0,0},

{0,0,0,0,0,2,0,1,6},

{8,11,0,0,0,0,1,0,7},

{0,0,2,0,0,0,6,7,0}};

dijkstra(graph,0);

return 0;

}

5)Take an example subnet of hosts and obtain a broadcast tree for the subnet.

/\*QUESTION-5: Subnet program\*/

#include<stdio.h>

int p,q,u,v,n;

int min=99;

int minCost=0;

int t[50][2],i,j;

int parent[50],edge[50][50];

int main(){

clrsrc();

printf("\nEnter the number of nodes");

scanf("%d",&n);

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

printf("%d",i);

parent[i]=-1;

}

printf("\n");

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

printf("%c",65+i);

for(j=0;j<n;j++){

scanf("%d",&edge[i][j]);

}

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

for(j=0;j<n;j++){

if(edge[1][j]!=99){

if(min<edge[i][j]){

min=edge[i][j];

u=i;

v=j;

}

p=find(u);

q=find(v);

if(p!=q){

t[i][0]=u;

t[i][1]=v;

minCost =minCost+edge[u][v];

sunion(p,q);

}

else{

t[i][0]=-1;

t[i][1]=-1;

}

}

}

}

min = 99;

}

printf("Minimum cost is %d\n Minimum spanning tree is\n",minCost);

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

if(t[i][0]!=-1 && t[i][1]!=1){

printf("%c %c %d",65+t[i][0],65+t[i][1],edge[t[i][0]] [t[i][1]]);

printf("\n");

}

getch();

}

sunion(int I,int m){

parent[] = m;

}

find(int I){

if(parent[1]>0){

I=parent[I];

}

return I;

}

}

6)Implement distance vector routing algorithm for obtaining routing tables at each node.

/\*QUESTION-6: Distance vector algorithm\*/

#include <stdio.h>

struct node {

unsigned dist[20];

unsigned from[20];

} rt[10];

int main() {

int costmat[20][20];

int nodes, i, j, k, count;

printf("Enter the number of nodes: ");

scanf("%d", &nodes);

printf("Enter the cost matrix:\n");

for (i = 0; i < nodes; i++) {

for (j = 0; j < nodes; j++) {

scanf("%d", &costmat[i][j]);

rt[i].dist[j] = costmat[i][j]; // Initialize distance

rt[i].from[j] = (costmat[i][j] != 0 && costmat[i][j] != 1000) ? j : -1; // Set 'from' based on cost

}

}

// Set diagonal to 0

for (i = 0; i < nodes; i++) {

rt[i].dist[i] = 0;

rt[i].from[i] = i;

}

// Floyd-Warshall Algorithm

do {

count = 0;

for (i = 0; i < nodes; i++) {

for (j = 0; j < nodes; j++) {

for (k = 0; k < nodes; k++) {

if (rt[i].dist[j] > rt[i].dist[k] + rt[k].dist[j]) {

rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j];

rt[i].from[j] = k;

count++;

}

}

}

}

} while (count != 0);

// Output the final routing table

printf("\nUpdated routing table\n");

for (i = 0; i < nodes; i++) {

for (j = 0; j < nodes; j++) {

if (rt[i].from[j] != -1) {

printf("%d ", rt[i].dist[j]);

} else {

printf("inf "); // Use "inf" to indicate no path

}

}

printf("\n");

}

return 0;

}

7)Implement data encryption and data decryption

/\*QUESTION-7:Encryption and decryption\*/

#include <stdio.h>

#include <string.h>

void encrypt(char \*text, int key) {

for (int i = 0; text[i] != '\0'; i++) {

// Encrypt uppercase letters

if (text[i] >= 'A' && text[i] <= 'Z') {

text[i] = (text[i] - 'A' + key) % 26 + 'A';

}

// Encrypt lowercase letters

else if (text[i] >= 'a' && text[i] <= 'z') {

text[i] = (text[i] - 'a' + key) % 26 + 'a';

}

}

}

void decrypt(char \*text, int key) {

// Decrypt by shifting in the opposite direction

for (int i = 0; text[i] != '\0'; i++) {

// Decrypt uppercase letters

if (text[i] >= 'A' && text[i] <= 'Z') {

text[i] = (text[i] - 'A' - key + 26) % 26 + 'A'; // +26 to handle negative

}

// Decrypt lowercase letters

else if (text[i] >= 'a' && text[i] <= 'z') {

text[i] = (text[i] - 'a' - key + 26) % 26 + 'a'; // +26 to handle negative

}

}

}

int main() {

char text[100];

int key;

printf("Enter the text to encrypt: ");

fgets(text, sizeof(text), stdin);

text[strcspn(text, "\n")] = 0; // Remove trailing newline

printf("Enter the encryption key (shift): ");

scanf("%d", &key);

encrypt(text, key);

printf("Encrypted text: %s\n", text);

decrypt(text, key);

printf("Decrypted text: %s\n", text);

return 0;

}