Structured Data Table

Name : Aadil Mohamed Puthiyaveetil

Reg No : 22BCE2436

Course Name: Computer Networks Lab

Course Code : BCSE308P

COMPUTER NETWORKS DA-2

Parity bit

```
#include <stdio.h>

int calculate_parity(unsigned char data, int parity_type) {
int parity = 0;
while (data) {
parity ^= (data & 1);
data >>= 1;
}
if (parity_type == 0) {
return parity;
} else {
return !parity;
}
```

```
}

void print_bits(unsigned char byte) {
for (int i = 7; i >= 0; i--) {
printf("%d", (byte >> i) & 1);
}
printf("\n");
}

int main() {
unsigned char data;
int parity_bit;
int parity_type;
unsigned char received_data;
printf("Aadil Mohamed Puthiyaveetil\n");
printf("Enter parity type (0 for even, 1 for odd): ");
scanf("%d", &parity_type);

if (parity_type != 0 && parity_type != 1) {
printf("Invalid parity type. Exiting.\n");
return 1;
}


printf("Enter an 8-bit data (0-255): ");
scanf("%hhu", &data);
```

```
parity_bit = calculate_parity(data, parity_type);

printf("Original data: ");
print_bits(data);

printf("Calculated parity bit: %d\n", parity_bit);

printf("Data with parity bit: ");
print_bits(data);
printf("%d (parity bit)\n", parity_bit);

printf("\nEnter the received 8-bit data (0-255): ");
scanf("%hhu", &received_data);

int received_parity_bit = calculate_parity(received_data, parity_type);

if (received_parity_bit == parity_bit) {
printf("No error detected.\n");
} else {
printf("Error detected in the data!\n");
}

return 0;
}
```

Output

CRC

```
#include <stdio.h>

#include <string.h>

#define MAX 100

void XOR(char dividend[], char divisor[], int n) {

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

if (dividend[i] == divisor[i]) {

dividend[i] = '0';

} else {

dividend[i] = '1';

}

}

}
```

```
void crc(char data[], char divisor[], char remainder[]) {
int dataLen = strlen(data);
int divisorLen = strlen(divisor);
char temp[MAX];

strcpy(temp, data);

for (int i = dataLen; i < dataLen + divisorLen - 1; i++) {
temp[i] = '0';
}
temp[dataLen + divisorLen - 1] = '\0';

for (int i = 0; i <= dataLen - 1; i++) {
if (temp[i] == '1') {
XOR(temp + i, divisor, divisorLen);
}
}

strcpy(remainder, temp + dataLen);
}

int main() {
char data[MAX], divisor[MAX], transmitted[MAX], remainder[MAX];

printf("Enter the data bits: ");
scanf("%s", data);
```

```
printf("Enter the divisor bits (generator polynomial): ");

scanf("%s", divisor);


crc(data, divisor, remainder);

printf("CRC remainder: %s\n", remainder);


strcpy(transmitted, data);

strcat(transmitted, remainder);

printf("Transmitted data with CRC: %s\n", transmitted);


char received[MAX];

printf("Enter received data: ");

scanf("%s", received);


crc(received, divisor, remainder);


int error = 0;

for (int i = 0; i < strlen(divisor) - 1; i++) {


if (remainder[i] != '0') {

error = 1;

break;

}

}


if (error) {
```

```
printf("Error detected in received data.\n");
} else {
printf("No error detected in received data.\n");
}

return 0;
}
```

OUTPUT

Hamming Code

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

void calculateParityBits(int data[], int n, int hammingCode[]) {
int r = 0;
while ((n + r + 1) > pow(2, r)) {
r++;
}

int parityIndex = 0, dataIndex = 0;
for (int i = 1; i <= n + r; i++) {
if ((i & (i - 1)) == 0) {
```

```
hammingCode[i - 1] = 0;

} else {

hammingCode[i - 1] = data[dataIndex++];

}

}

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

int parityPos = pow(2, i);

int count = 0;

for (int j = parityPos; j <= n + r; j += (2 * parityPos)) {

for (int k = j; k < j + parityPos && k <= n + r; k++) {

if (hammingCode[k - 1] == 1) {

count++;

}

}

}

hammingCode[parityPos - 1] = count % 2;

}

}

void printArray(int arr[], int size) {

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

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

}
```

```
printf("\n");

}

int main() {

int n;

printf("Aadil Mohamed Puthiyaveetil\n");

printf("22BCE2436\n");

printf("Enter the number of data bits: ");

scanf("%d", &n);

int data[n];

printf("Enter the data bits: ");

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

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

}

int r = 0;

while ((n + r + 1) > pow(2, r)) {

r++;

}

int hammingCode[n + r];

calculateParityBits(data, n, hammingCode);

printf("Calculated Hamming code: ");
```

```
printArray(hammingCode, n + r);

int userCode[n + r];

printf("Enter the Hamming code for verification: ");

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

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

}

int isMatching = 1;

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

if (userCode[i] != hammingCode[i]) {

isMatching = 0;

break;

}

}

if (isMatching) {

printf("The entered Hamming code matches the calculated code.\n");

} else {

printf("The entered Hamming code does not match the calculated

code.\n");

}

return 0;

}
```

OUTPUT

Checksum

```
#include <stdio.h>

int calculateChecksum(int data[], int length) {
int sum = 0;
for (int i = 0; i < length; i++) {
sum += data[i];
if (sum > 0xFFFF) {
sum = (sum & 0xFFFF) + (sum >> 16);
}
}
return ~sum & 0xFFFF;
}
```

```
int main() {

printf("Name: Aadil Mohamed\n");

printf("Reg No: 22BCE2436\n");

int n;

printf("Enter the number of data blocks: ");

scanf("%d", &n);

int data[n + 1];


printf("Enter the data blocks (16-bit hexadecimal values, e.g., 0xABCD):\n");

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

scanf("%x", &data[i]);

}


int checksum = calculateChecksum(data, n);

printf("Calculated checksum: 0x%X\n", checksum);


data[n] = checksum;

printf("Transmitting data with checksum: ");

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

printf("0x%X ", data[i]);

}

printf("\n");


int receivedData[n + 1];

printf("Enter received data with checksum: ");
```

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

scanf("%x", &receivedData[i]);

}


int receiverChecksum = calculateChecksum(receivedData, n + 1);

if (receiverChecksum == 0xFFFF) {

printf("Data received correctly.\n");

} else {

printf("Error detected in data transmission.\n");

}
```

Output