TRIPLE C-Code Prototype

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

#define MAX_TRIPL_LENGTH 256  // One entry per ASCII character
#define OBFUSCATION_MASK 0xAB // XOR mask for reversible transformation

typedef struct {
    char character;
    char encoded_value[5]; // e.g., "x5F"
} TRIPLPointer;

typedef struct {
    TRIPLPointer map[MAX_TRIPL_LENGTH];
    int count;
    time_t start_time;
} TRIPLArray;

// Build deterministic ASCII map with encoded representations
TRIPLArray create_tripl_array() {
    TRIPLArray array;
    array.count = MAX_TRIPL_LENGTH;
    array.start_time = time(NULL);

    for (int i = 0; i < MAX_TRIPL_LENGTH; i++) {
        array.map[i].character = (char)i;
        snprintf(array.map[i].encoded_value,
sizeof(array.map[i].encoded_value), "x%02X", i ^ OBFUSCATION_MASK);
    }

    return array;
}

// Encrypt using direct ASCII index (deterministic mapping)
void encrypt_message(const char *message, TRIPLArray *array, char
**encrypted) {
    int i = 0;
    while (message[i] != '\0') {
        unsigned char index = (unsigned char)message[i];
        encrypted[i] = strdup(array->map[index].encoded_value);
        i++;
    }
    encrypted[i] = NULL; // Mark end of encrypted sequence
}

// Decrypt using reverse lookup (linear scan)
void decrypt_message(char **encrypted, TRIPLArray *array, char *output) {
```

```
    int i = 0;
    while (encrypted[i] != NULL) {
        for (int j = 0; j < array->count; j++) {
            if (strcmp(encrypted[i], array->map[j].encoded_value) == 0) {
                output[i] = array->map[j].character;
                break;
            }
        }
        i++;
    }
    output[i] = '\0'; // Proper null-termination
}

// Demonstration
int main() {
    TRIPLArray tripl = create_tripl_array();
    const char *original = "HELLO TRIPLE!";
    char *encrypted[1024];
    char decrypted[1024];

    encrypt_message(original, &tripl, encrypted);
    printf("Encrypted:\n");
    for (int i = 0; encrypted[i] != NULL; i++) {
        printf("%s ", encrypted[i]);
    }
    printf("\n");

    decrypt_message(encrypted, &tripl, decrypted);
    printf("Decrypted: %s\n", decrypted);

    for (int i = 0; encrypted[i] != NULL; i++) {
        free(encrypted[i]);
    }

    return 0;
}
```