PROJE RAPORU

1. TOPLANTI: İlk toplantımızı 16 Aralık 2024 tarihinde discord üzerinden gerçekleştirdik. Toplantıda ödev konumuza baktık ve üstünkörü bir iş dağılımı yaptık. Ödevi yaparken nasıl adımlardan geçerek tamamlayacağımızı kararlaştırıp toplantımızı sona erdirdik.
2. TOPLANTI: İlk toplantımızda kararlaştırdığımız adımları uygulayarak ilk başta istişare yaparak kodumuzu yazdık, ekran çıktılarını aldık sonrasında akış şemamızı oluşturduk. Kodu Eren’in bilgisayarı üzerinden yaptığımız için rapor yazma kısmını Eren’e bıraktık. Bu şekilde toplantılarımız sona erdi.

ÖDEV KONUSU: Bir askeri kurumda çalışanlar bina girişlerinde elektronik kartlarla giriş yapmaktadırlar. Düşman ülkenin casuslarından birisi kurum çalışanlarından birinin kartını kopyalayarak kuruma sızmayı başarmıştır. Müfettişler sızmayı ortaya çıkarabilmek için bina girişinde çalışanların kimlik numaralarının kaydedildiği dosyayı tarayarak gün içinde birbirinin aynı olan iki adet kimlik numarasının olup olmadığını araştırmak istemektedirler. İçinde kimlik numaralarını barındıran "giris.txt" adlı bir metin (text) dosyasını tarayarak, birden fazla giriş yapmış kimlik numarasını "casus .txt" dosyasına kaydeden programı yazınız. NOT: Giriş dosyasının içindeki kayıt sayısı bilinmemektedir.

C Program Kodu:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_KIMLIK 20 // Kimlik numarasının maksimum uzunluğunu 20 karakter olarak belirledik

#define HASH\_SIZE 1000 // Hash tablosunun boyutunu 1000 olarak belirledik

// Kimlik numaralarını saklamak için kullanılan bağlı liste yapısını tanımlıyoruz

typedef struct Node {

char kimlik[MAX\_KIMLIK]; // Her bir kimlik numarasını saklamak için bir dizi

struct Node\* next; // Bir sonraki kimlik numarasını işaret eden işaretçi

} Node;

// Hash fonksiyonu: Kimlik numarasını alıp, hash tablosundaki uygun yuvayı buluyor

unsigned int hash(char \*kimlik) {

unsigned int hashValue = 0;

while (\*kimlik) {

// Kimlik numarasındaki her karakteri kullanarak hash değeri oluşturuyoruz

hashValue = (hashValue << 5) + \*kimlik++;

}

// Oluşturulan hash değerini tablonun boyutuna göre sınırlıyoruz

return hashValue % HASH\_SIZE;

}

// Kimlik numarasını hash tablosuna eklemek için bu fonksiyonu kullanıyoruz

int addKimlik(Node\* hashTable[], char\* kimlik) {

unsigned int index = hash(kimlik); // Kimlik için uygun indexi buluyoruz

Node\* current = hashTable[index]; // O indexteki ilk elemanı kontrol ediyoruz

// Eğer kimlik daha önce tablodaysa, bu bir casus kimliktir

while (current != NULL) {

if (strcmp(current->kimlik, kimlik) == 0) {

return 1; // Kimlik zaten eklenmişse 1 döndürüyoruz, yani casus tespit edilmiştir

}

current = current->next; // Eğer değilse, bağlı listeyi devam ettiriyoruz

}

// Eğer kimlik daha önce eklenmemişse, yeni bir node oluşturup ekliyoruz

Node\* newNode = (Node\*)malloc(sizeof(Node)); // Yeni bir node alıyoruz

strcpy(newNode->kimlik, kimlik); // Yeni kimlik numarasını node'a ekliyoruz

newNode->next = hashTable[index]; // Yeni node'u mevcut başa ekliyoruz

hashTable[index] = newNode; // Tabloya ekliyoruz

return 0; // Yeni kimlik eklenmişse 0 döndürüyoruz

}

int main() {

FILE \*girisDosyasi, \*casusDosyasi; // Dosyaları tanımlıyoruz

char kimlik[MAX\_KIMLIK]; // Her bir kimlik numarasını tutmak için bir dizi

Node\* hashTable[HASH\_SIZE] = {0}; // Hash tablosunu başlatıyoruz (tüm elemanlar NULL)

// Giriş dosyasını açıyoruz, eğer açılmazsa hata mesajı verir

girisDosyasi = fopen("giris.txt", "r");

if (girisDosyasi == NULL) {

perror("Giris dosyasi acilamadi"); // Dosya açılmadığında sistem hatasını gösterir

return 1; // Hata durumunda programı sonlandırıyoruz

}

// Casus kimliklerin kaydedileceği dosyayı açıyoruz

casusDosyasi = fopen("casus.txt", "w");

if (casusDosyasi == NULL) {

perror("Casus dosyasi acilamadi"); // Casus dosyası açılamazsa hata mesajı verir

fclose(girisDosyasi); // Giriş dosyasını kapatıyoruz

return 1; // Hata durumunda programı sonlandırıyoruz

}

// Giriş dosyasındaki her bir kimlik numarasını okuyoruz

while (fgets(kimlik, sizeof(kimlik), girisDosyasi)) {

kimlik[strcspn(kimlik, "\n")] = 0; // Kimlik numarasının sonunda bulunan yeni satır karakterini temizliyoruz

// Eğer kimlik daha önce kaydedildiyse, bu bir casus kimliktir

if (addKimlik(hashTable, kimlik)) {

fprintf(casusDosyasi, "%s\n", kimlik); // Casus kimlikleri kaydediyoruz

}

}

// Dosyaları kapatıyoruz

fclose(girisDosyasi); // Giriş dosyasını kapatıyoruz

fclose(casusDosyasi); // Casus dosyasını kapatıyoruz

// Hash tablosunda kullanılan hafızayı temizliyoruz

for (int i = 0; i < HASH\_SIZE; i++) {

Node\* current = hashTable[i]; // Bu indexteki tüm kimlikleri kontrol ediyoruz

while (current != NULL) {

Node\* temp = current;

current = current->next;

free(temp); // Hafızayı serbest bırakıyoruz

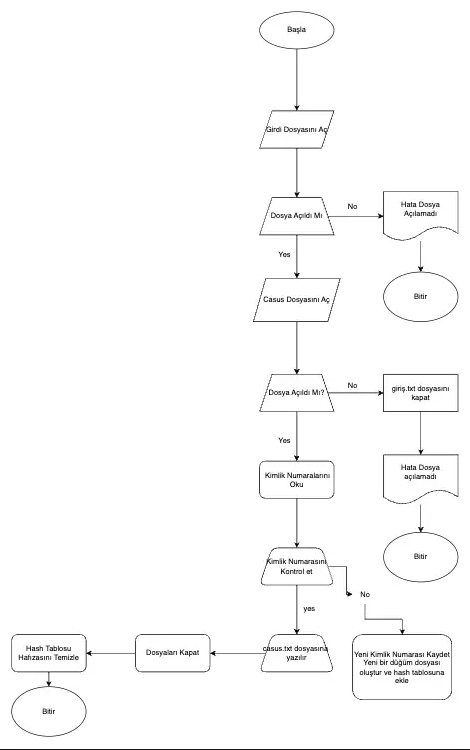
}

}

// Casus kimliklerin kaydedildiğini bildiriyoruz

printf("Casus kimlikler kaydedildi.\n");

return 0; // Programı başarıyla bitiriyoruz  
}

AKIŞ ŞEMASI:  
  
  
  
  
  
  
  
  
  
  
  
  
  
EKRAN ÇIKTILARI:  
