PROGRAMLAMA – II Dönem Ödevi

NumTool – Sayı Teorisi Hesaplama Aracı

1. Ödevin Amacı

Bu ödevde öğrencilerden C programlama dili kullanarak komut tabanlı bir sayı teorisi hesaplama aracı geliştirmeleri beklenmektedir. Program bir giriş dosyasından komutları okuyacak, ayrıştıracak, dinamik veri yapıları içerisinde saklayacak ve ilgili algoritmaları çalıştırarak sonuçları bir çıktı dosyasına yazacaktır. Bu ödev yalnızca matematiksel işlemlerin doğruluğunu değil; dinamik bellek yönetimi (pointer kullanımı), modüler programlama (çoklu dosya mimarisi), Makefile otomasyonu ve gelişmiş metin ayrıştırma (string parsing) becerilerinin ölçülmesini amaçlamaktadır.

2. Programın Genel Çalışma Yapısı

Program komut satırından çalıştırılacak ve tüm işlemler modüler bir mimari üzerinde yürütülecektir.

2.1. Programın Çalıştırılması ve Komut Satırı Argümanları (CLI)

Programınız doğrudan bir IDE üzerinden değil, komut satırından çalıştırılacak şekilde tasarlanmalıdır.

- **Beklenen Kullanım Formatı:**
  - Linux/macOS: `./numtool <giris_dosyasi> <cikis_dosyasi>`
  - Windows: `.\numtool[.exe] <giris_dosyasi> <cikis_dosyasi>`
- **Önemli Kural:** `main` fonksiyonu ilk iş olarak doğru sayıda argüman girilip girilmediğini kontrol etmelidir (Örn: `if (argc != 3)`). Hatalı kullanımda kullanıcıya bilgi mesajı verilip program sonlandırılmalıdır.

2.2. Modüler Kod Mimarisi ve Makefile (ZORUNLU)

Tüm kodun tek bir `main.c` dosyasına yazılması **kesinlikle yasaktır**. Projenizi mantıksal modüllere ayırmanız gerekmektedir. Örnek bir mimari şu şekilde olmalıdır:

- `main.c`: Sadece argüman kontrolünü, dosyaların açılıp kapanmasını ve ana döngüyü (kontrol akışını) yönetir.
- `parser.c` ve `parser.h`: Dosya okuma, boşlukları temizleme ve metin ayrıştırma (strtok/sscanf) işlemlerini barındırır.
- `math_utils.c` ve `math_utils.h`: İstenen sayı teorisi algoritmalarını (GCD, PRIME, vb.) içerir.
- **Makefile:** Projenin derlenmesi manuel olarak `gcc` komutlarıyla yapılmamalıdır. Bir `Makefile` yazmalısınız. Terminalde sadece `make` komutu çalıştırıldığında tüm modüller derlenip `numtool` adlı çalıştırılabilir dosya (executable) otomatik olarak üretilmelidir.
- Makefile platform karmaşalarının önlenmesi için, Windows kullanıcıları makefile'larını **WSL (Windows Subsystem for Linux)** aracılığıyla kuracakları bir Linux işletim sisteminde(örn: Ubuntu) oluşturmalıdır.

3. Girdi Dosyası Kuralları ve Ayrıştırma (Parsing)

Programınızın sağlam bir parser kullanarak aşağıdaki durumları doğru ele alması gerekmektedir:

- Satır başında, sonunda veya komutlar arasında gereksiz/fazla boşluklar olabilir.
- Boş satırlar bulunabilir. # karakteri ile başlayan satırlar yorum satırı kabul edilip atlanmalıdır.
- Aynı satırda birden fazla komut bulunabilir, bu komutlar noktalı virgül (;) ile ayrılır.

İpucu: Dosyadan güvenli satır okumak için fgets, aynı satırdaki komutları bölmek için strtok, ve bir komutun içinden parametreleri çekmek için sscanf fonksiyonlarını araştırmanız işinizi çok kolaylaştıracaktır.

4. Komut ve Parametre Tablosu

Programınız aşağıdaki komutları desteklemeli ve belirtilen hata durumlarını yönetmelidir:

Komut	Parametreler	İşlem Özeti	Başarılı Çıktı Formatı	Hata Çıktısı
GCD	A B	A ve B'nin EBOB'unu hesaplar.	GCD A B -> Sonuç	GCD A B -> ERROR_INVALID_INPUT (Negatif/Sıfır ise)
POW	Taban Üs Mod	Taban^Üs mod Mod hesaplar.	POW Taban Üs Mod -> Sonuç	POW Taban Üs Mod -> ERROR_INVALID_INPUT
PRIME	N	N'in asal olup olmadığını kontrol eder.	PRIME N -> YES / NO	PRIME N -> ERROR_INVALID_INPUT (N < 2 ise)
INV	A M	A'nın M modundaki tersini hesaplar.	INV A M -> Sonuç	INV A M -> ERROR_NO_INVERSE (Aralarında asal değilse)
PHI	N	Euler Totient fonksiyonunu hesaplar.	PHI N -> Sonuç	PHI N -> ERROR_INVALID_INPUT (N < 1 ise)
CHECK	A M	Modüler tersin doğruluğunu test eder.*	CHECK A M -> CORRECT	CHECK A M -> FAILED veya ERROR_NO_INVERSE

(**CHECK Komutu İşleyişi:** A'nın M modundaki tersi hesaplanır. Sonuç A ile çarpılıp M'ye göre modu alınır. Sonuç 1 çıkıyorsa CORRECT basılır.)

5. Dinamik Veri Yapısı Tasarımı (ZORUNLU)

Her komutu ve sonucunu saklamak için aşağıdaki gibi bir veri yapısı kullanmalısınız:

```
typedef struct {
    char command[20];
    long long args[4];
    int arg_count;
    char result[100];
    int is_error; // Hata oluştuğunda 1 yapılır.
} CommandRecord;
```

Kritik Kural: Girdi dosyasında kaç adet komut olduğu önceden bilinmemektedir. Bu nedenle CommandRecord records[1000]; gibi **sabit boyutlu (static) diziler kullanmak kesinlikle yasaktır.** Bunun yerine:

1. Ya malloc ve realloc kullanarak dizi boyutunu okunan satır sayısına göre dinamik olarak genişleteceksiniz (Dinamik Dizi).
2. Ya da komutları bir **Bağlı Liste (Linked List)** yapısı kullanarak bellekte tutacaksınız. Programınızın işi bittiğinde, ayırdığınız tüm dinamik belleği free() fonksiyonu ile sisteme geri iade etmeniz (Memory Leak oluşturmamanız) en önemli değerlendirme kriterlerinden biridir.

6. Kullanılması Beklenen Algoritmalar ve Teknik İsterler

Proje kapsamında kullanılacak algoritmaların, C standart kütüphanesindeki hazır matematik fonksiyonları (<math.h> vb.) kullanılmadan, aşağıda belirtilen matematiksel altyapıya ve zaman karmaşıklığı (Time Complexity) kısıtlarına uygun şekilde implement edilmesi zorunludur.

6.1. Öklid Algoritması (GCD)

- **Matematiksel Tanım:** GCD(a, b) = GCD(b, a % b)
- **Teknik İster:** Algoritma O(log(min(a, b))) zaman karmaşıklığında çalışmalıdır. Ardışık modül (%) alma işlemi uygulanarak b = 0 sınır koşuluna ulaşıldığında a değeri döndürülmelidir.

6.2. Genişletilmiş Öklid Algoritması ve Modüler Ters (INV)

- **Matematiksel Tanım**: a*x + b*y = GCD(a, b) denklemini sağlayan x ve y katsayılarını hesaplar.
- **Modüler Ters İsteri:** (A * x) % M = 1 eşitliğindeki x değerini (modüler ters) bulmak için kullanılmalıdır. Algoritma O(log(min(a, m))) sürede çalışmalıdır.
- **Hata Kontrolü:** Eğer GCD(a, m) != 1 ise tersin olmadığı belirtilmelidir (ERROR_NO_INVERSE). Algoritma sonucunda katsayı (x) negatif çıkarsa, doğru pozitif modüler değer x = (x % M + M) % M formülü ile bulunmalıdır.

6.3. İkili Üs Alma (Binary Exponentiation - POW)

- **Matematiksel Tanım:** (a^b) % m işleminin hızlı hesaplanması metodudur.
- **Teknik İster:** Standart O(b) karmaşıklığındaki döngülü çarpım zaman aşımına neden olacağından, algoritma kesinlikle O(log b) zaman karmaşıklığına sahip olmalıdır. Üs değeri (b) ikiye bölünerek (veya >> 1 ile kaydırılarak) incelenmeli, üssün tek olduğu durumlarda sonuç güncellenirken, her adımda taban değerinin karesi alınmalıdır.
- **Taşma (Overflow) Koruması:** long long çarpımlarında bellek taşmasını önlemek için, her çarpım işleminden hemen sonra % m işlemi uygulanmalıdır.

6.4. Asallık Testi (PRIME)

- **Teknik İster:** Asallık testi en kötü durumda O(sqrt(N)) zaman karmaşıklığında çalışmalıdır.
- **Optimizasyon Şartı:** N <= 1 reddedilmeli, N=2 ve N=3 kabul edilmelidir. Çift sayılar ve 3'ün katları baştan elendikten sonra, döngü i = 5'ten başlayarak i <= sqrt(N) sınırına kadar i += 6 adımlamasıyla (sadece 6k - 1 ve 6k + 1 formundaki sayıları bölerek) yapılmalıdır.

6.5. Euler Totient Fonksiyonu (PHI)

- **Matematiksel Tanım:** PHI(N) = N * (1 - 1/p1) * (1 - 1/p2) * ... * (1 - 1/pk) (Burada p değerleri sayının asal çarpanlarıdır).
- **Teknik İster**: O(sqrt(N)) zaman karmaşıklığında asal çarpanlara ayırma yöntemi kullanılmalıdır.
- **Veri Tipi Hassasiyeti**: Formüldeki kesirli hesaplamalar float/double ile yapıldığında hassasiyet kaybı (precision loss) yaşanır. Bunu önlemek için sadece tam sayı bölmesi kullanılarak sonuc = sonuc - (sonuc / p) mantığıyla koda dökülmelidir.

7. Örnek Girdi ve Çıktı (Kapsamlı Test Senaryoları)

Aşağıdaki örnekler programınızın hem temel vakalarda hem hata durumlarında hem de performans testlerinde nasıl çalışması gerektiğini göstermektedir. Projeniz değerlendirilirken bu büyüklükteki sayılarla test edilecektir.

Girdi Örneği (`input.txt`):

Plaintext

```
# --- TEMEL TESTLER ---
GCD 48 18; POW 2 10 1000
PRIME 29
INV 3 11; PHI 36

# --- HATA KONTROLLERİ ---
INV 4 8
PRIME 1
GCD -5 10

# --- BÜYÜK SAYILAR VE PERFORMANS TESTİ ---
GCD 987654321 123456789
POW 2 30 1000000007
PRIME 2147483647
PHI 1000000007
CHECK 1234567 1000000007
```

Beklenen Çıktı Formatı (`output.txt`):

Plaintext

```
GCD 48 18 -> 6
POW 2 10 1000 -> 24
PRIME 29 -> YES
INV 3 11 -> 4
PHI 36 -> 12
INV 4 8 -> ERROR_NO_INVERSE
PRIME 1 -> ERROR_INVALID_INPUT
GCD -5 10 -> ERROR_INVALID_INPUT
GCD 987654321 123456789 -> 9
POW 2 30 1000000007 -> 73741817
PRIME 2147483647 -> YES
PHI 1000000007 -> 1000000006
CHECK 1234567 1000000007 -> CORRECT
```

8. Değerlendirme Kriterleri

Projenin puanlanmasında dikkate alınacak kriterler aşağıda verilmiştir.

- Algoritmaların Doğruluğu ve Optimizasyonu – %20
- Dinamik Bellek Yönetimi (malloc/free kullanımı) – %10
- Modüler Mimari ve Makefile Kurulumu – %10
- Girdi Dosyasının Doğru Ayrıştırılması (Parsing) – %15
- Hatalı Komutların / İstisnaların Ele Alınması – %5
- Koda hakimiyet- %30
- Performans (Zaman aşımına uğramama) ve CHECK komutu – %5
- Kod Düzeni, Test Dosyası ve Proje Raporu – %5

Kaynak Kod Yazımı ve Rapor Gönderimi

● Kaynak kod yazarken kaynak kodun en başına açıklama satırı şeklinde, ad, soyad, numara bilgileri yazılmalıdır.

● Gerekli yerlere açıklama satırları eklenmeli ve kodun okunabilirliğini sağlamak için girintilere ve değişken isimlerine dikkat edilmelidir.

● Ayrıca kaynak kodun görsel açıdan hizalı olmasına dikkat edilmelidir.

● Ödev teslimi için kaynak kodları çalışır halde bütün modüller, makefile ve ödev teslim raporu ile birlikte bir klasörde sıkıştırarak (.zip, .7z, .rar, .tar.gz, vs.) gönderilmelidir.

● Ödev savunmaları esnasında öğrenciden bu dosyadaki kaynak kodlar kullanılarak çalıştırılabilir duruma getirilmesi ve değişiklik yapması istenebilecektir.