1. İŞLEÇ ALIŞTIRMALARI (1)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int y;

    y = -2 + 5 * 3 - 4;
    printf("y = %d\n", y);

    y = 4 + 5 % 5 - 5;
    printf("y = %d\n", y);

    y = (6 + 7) % 5 / 2;
    printf("y = %d\n", y);

    return 0;
}
```

****

2. İŞLEÇ ALIŞTIRMALARI (2)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 10;

    printf("%d\n", x++);
    printf("%d\n", ++x);
    printf("%d\n", x++);
    printf("%d\n", ++x);
    printf("%d\n", x);

    return 0;
}
```

****

3. İŞLEÇ ALIŞTIRMALARI (3)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 10;
    int y = 5;

    printf("%d\n", ++x - y--);
    printf("%d\n", x++ + --y);
```

```
    printf("%d\n", ++x + y--);
    printf("%d\n", x - y);

    return 0;
}
```

****

4. İŞLEÇ ALIŞTIRMALARI (4)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = -1;
    int y = 0;

    printf("%d\n", !++x - !y--);
    printf("%d\n", !++x - !y--);
    printf("%d\n", ++x + y);

    return 0;
}
```

****

5. İŞLEÇ ALIŞTIRMALARI (5)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 5;
    int y, z;

    x *= 3 + 1;
    printf("x = %d\n", x);

    x *= y = z = 3;
    printf("x = %d\n", x);

    x = y == z;
    printf("x = %d\n", x);

    x == (y = z);
    printf("x = %d\n", x);

    return 0;
}
```

****

6. İŞLEÇ ALIŞTIRMALARI (6)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 3, y = 2, z = 0;

    x = x && y || z;
    printf("x = %d\n", x);

    printf("%d\n", x || !y && z);

    x = y = 1;
    z = x++ - 1;
    printf("x = %d\n", x);
    printf("z = %d\n", z);

    z += -x++ + ++y;
    printf("x = %d\n", x);
    printf("z = %d\n", z);

    return 0;
}
```

****

7. İŞLEÇ ALIŞTIRMALARI (7)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 7;
    int y = 5;
    int z;

    z = x+++y;
    y = z---x;
    x = y+++z;

    printf("x = %d\n", x);
    printf("y = %d\n", y);
    printf("z = %d\n", z);

    return 0;
}
```

****

8. İŞLEÇ ALIŞTIRMALARI (8)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 1, y = 1, z = 1;
```

```
    ++x || ++y && ++z;
    printf("%d %d %d\n", x, y, z);

    x = y = z = 1;
    ++x && ++y || ++z;
    printf("%d %d %d\n", x, y, z);

    x = y = z = 1;
    ++x && ++y && ++z;
    printf("%d %d %d\n", x, y, z);

    x = y = z = -1;
    ++x && ++y || ++z;
    printf("%d %d %d\n", x, y, z);

    x = y = z = -1;
    ++x || ++y && ++z;
    printf("%d %d %d\n", x, y, z);

    x = y = z = -1;
    ++x && ++y && ++z;
    printf("%d %d %d\n", x, y, z);

    return 0;
}
```

****

9. ÜÇ BASAMAKLI SAYININ TERSİ

***x*, *int* türden, içinde *3* basamaklı bir deger tutan bir degişken olsun.**
Öyle bir ifade yazın ki, ifadenin değeri *x* değerinin tersi olsun (örneğin *345* için *543*).

```
#include <stdio.h>

int main()
{
    int x;

    printf("3 basamaklı bir sayı girin : ");
    scanf("%d", &x);

    printf("%d ----> %d\n", x, /* ifade */);

    return 0;
}
```

****

10. if DEYİMİ ALIŞTIRMALARI (1)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int x = 0;
    int y = 1;
    int z = 1;
```

```
    if (x = 0)
        y = 0;

    if (y == 0)
        z = 0;

    if (z == 1);
        x = 1;

    printf("x = %d\n", x);
    printf("y = %d\n", y);
    printf("z = %d\n", z);

    return 0;
}
```

****

11. if DEYİMİ ALIŞTIRMALARI (2)

```
#include <stdio.h>

int main()
{
    int x, y = 1, z;

    if (y) x = 5;
    printf("x = %d\n", x);

    if (!y)
        x = 3;
    else
        x = 5;
    printf("x = %d\n", x);

    x = 1;
    if (y < 0)
        if (y > 0)
            x = 3;
    else
        x = 5;
    printf("x = %d\n", x);

    if (z = y < 0)
        x = 3;
    else if (y == 0)
        x = 5;
    else
        x = 7;
    printf("x = %d\n", x);
    printf("z = %d\n", z);
    if (x = z = y);
        x = 3;

    printf("x = %d\n", x);
    printf("z = %d\n", z);

    return 0;
}
```

****

12. if DEYİMİ ALIŞTIRMALARI (3)

```
#include <stdio.h>

int main()
{
    int i = 6;

    if (((++i < 7) && ( i++/6)) || (++i <= 9))
        ;

    printf("%d\n",i);

    return 0;
}
```

****

13. İKİ SAYIDAN BÜYÜĞÜNÜ BULAN İŞLEV

Kendisine gönderilen *int* türden iki sayının büyük olanına geri dönen, *get_max2* isimli işlevi tanımlayın:

```
int get_max2(int number1, int number2, int number3);
```

Yazmış olduğunuz işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

int get_max2(int number1, int number2);

int main()
{
    int x, y;

    printf("iki sayı giriniz : ");
    scanf("%d%d ", &x, &y);
    printf("%d ve %d icin max = %d\n", x, y, get_max2(x, y));

    return 0;
}
```

****

14. ÜÇ SAYIDAN BÜYÜĞÜNÜ BULAN İŞLEV

Kendisine gönderilen *int* türden üç sayının en büyük olanına geri dönen, *get_max3* isimli işlevi tanımlayın:

```
int get_max3(int number1, int number2, int number3);
```

Yazmış olduğunuz işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

int get_max3(int number1, int number2, int number3);

int main()
{
    int s1, s2, s3;
```

```
    printf("üç sayı giriniz : ");
    scanf("%d%d%d", &s1, &s2, &s3);
    printf("%d %d %d için max = %d\n", s1, s2, s3, get_max3(s1, s2, s3));

    return 0;
}
```

****

15. ÜÇ SAYIDAN ORTANCASINI BULAN İŞLEV

Kendisine gönderilen üç sayıdan ortancasına geri dönen, *get_mid* isimli işlevi yazın. Eğer gönderilen değerlerden herhangi ikisi ya da her üçü aynı ise, işlev bu değere geri dönmeli. Yazmış olduğunuz işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

int get_mid(int, int, int);

int main()
{
    int x, y, z;

    printf("Üç sayı giriniz : ");
    scanf("%d%d%d", &x, &y, &z);
    printf("ortanca sayı = %d\n", get_mid(x, y, z));

    return 0;
}
```

****

16. CELCİUS'DAN FAHRENHEİT'A

Kendisine gönderilen *Celsius* degerinin Fahrenheit eşdeğeri ile geri dönen *cel_to_fahr* isimli işlevi yazın:

```
int cel_to_fahr(int celsius);
```

Yazılan işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int cel_to_fahr(int celsius);

int main()
{
    int cel;

    printf("celcius degerini giriniz: ");
    scanf("%d", &cel);
    printf("%d celsius = %d fahrenheit\n", cel, cel_to_fahr(cel));

    return 0;
}
```

****

17. SİGNUM İŞLEVİ

Kendisine gönderilen *int* türden bir tamsayının negatif mi, *0* mı, pozitif mi olduğunu sınayan *signum* isimli işlevi tanımlayın:

```
int signum(int value);
```

İşlev

value == 0 olması durumunda 0 değerine
value < 0 olması durumunda -1 değerine
value > 0 olması durumunda 1 değerine

geri dönmeli.

Yazılan işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int signum(int value);

int main()
{
    int a;

    printf("bir sayı girin : ");
    scanf("%d", &a);

    if (signum(a) == 1)
        printf("Girdiginiz sayı pozitif bir sayıdır\n");
    else if (signum(a) == -1)
        printf("Girdiginiz sayı negatif bir sayıdır\n");
    else
        printf("Girdiginiz sayı 0'dır\n");

    return 0;
}
```

****

18. SÜRE GÖRÜNTÜLEME

Kendisine gönderilen saniye cinsinden süreyi saat, dakika ve saniye olarak ayrıştırarak ekrana yazdıran *display_duration* işlevini yazın. Eğer saat, dakika ya da saniye değeri *0* ise, bu değer ekrana yazdırılmamalı.

```
void display_duration(int sec);
```

Yazmış olduğunuz işlev aşağıdaki kod ile sınayabilirsiniz:

```
#include <stdio.h>

void display_duration(int sec);

int main()
{
    display_duration(3600);
    display_duration(240);    display_duration(310);
    display_duration(7205);   display_duration(14520);
    display_duration(4212);
```

```
    return 0;
}
```

****

19. BASAMAK DEĞERİNDEN KARAKTERE

Onaltılık sayı sisteminde basamak değeri olan bir tamsayıya ilişkin karakterin kod numarasını döndüren *get_hex_char* isimli işlevi yazın.

```
int get_hex_char(int)
```

***get_hex_char* işlevi kendisine gönderilen *int* türden bir değeri onaltılık sayı sistemine ait bir basamak değeri olarak ele alarak, bu basamak değerine ilişkin karakterin kod numarasına geri dönmeli. Geri dönüş değeri büyük harf karakterlerine ilişkin olmalı. Yazmış olduğunuz işlev aşağıdaki kod ile sınayabilirsiniz:**

```
#include <stdio.h>

int get_hex_char(int);

int main()
{
    putchar(get_hex_char(15));
    putchar(get_hex_char(10));
    putchar(get_hex_char(0));
    putchar(get_hex_char(5));
    putchar(get_hex_char(13));

    return 0;
}
```

****

20. KARAKTERDEN BASAMAK DEĞERİNE

Onaltılık sayı sisteminde bir basamağa ilişkin karakterin (kullanılan karakter setindeki) kod numarasını alarak bu basamağa ilişkin basamak değerine geri dönen *get_hex_value* işlevini yazın:

```
int get_hex_value(int ch)
```

İşlevin parametre değişkenine, onaltılık sayı sisteminde bir basamak değeri gösteren karakterin sıra numarası geçildiğinde, işlev bu basamağa ilişkin basamak değeriyle geri dönmeli. İşleve onaltılık sayı sisteminde bir basamak değeri olamayacak bir karakterin sıra numarası gönderilirse, işlev *-1* değerine geri dönmeli. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F ,a, b, c, d, e, f, geçerli karakterlerdir.
Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int get_hex_value(int);

int main()
{
    printf("%d\n", get_hex_value('a'));       /* 10 */
    printf("%d\n", get_hex_value('6'));       /* 6  */
    printf("%d\n", get_hex_value('F'));       /* 15 */
    printf("%d\n", get_hex_value('0'));       /* 0  */
    printf("%d\n", get_hex_value('C'));       /* 12 */
```

```
    printf("%d\n", get_hex_value('M'));       /* -1 */

    return 0;
}
```

****

21. ÜÇGEN ÇİZİLEBİLİR Mİ?

Üç kenar uzunlugu verilen üçgenin çizilip çizilemiyeceğini sınayan, eğer çizilebiliyorsa sıfır dışı değerle, çizilemiyorsa *0* değeri ile geri dönen, *is_triangle* isimli işlevi yazın. İşlevin bildirimi aşağıdaki gibidir:

```
int is_triangle(unsigned int a, unsigned int b, unsigned int c);
```

Yazmış olduğunuz işlev aşağıdaki kod ile sınayabilirsiniz:

```
#include <stdio.h>

int is_triangle(unsigned int a, unsigned int b, unsigned int c);

int main()
{
    int a, b, c;

    printf("ABC üçgeni için kenar uzunluklarını giriniz.\n");
    scanf("%d%d%d", &a, &b, &c);

    if (is_triangle(a, b, c))
        printf("ABC üçgeni çizilebilir.\n");
    else
        printf("ABC üçgeni çizilemez.\n");

    return 0;
}
```

****

22. ARTIK YIL SINAMASI

4'e tam bölünen yıllar içinde *100*'e tam bölünmeyenler ile *400*'e tam bölünenler, artık yıldır. Birkaç örnek:
***1967* artık yıl değil. (*4*'e tam bölünmüyor.)**
1964 artık yıl. (4'e tam bölünüyor ve 100'e tam bölünmüyor.)
2000 artık yıl. (4'e tam bölünüyor ve 400'e tam bölünmüyor.)
1900 artık yıl değil. (4'e tam bölünüyor, 100'e tam bölünüyor ve 400'e tam bölünmüyor.)
Kendisine gönderilen bir yılın, artık yıl olup olmadığını sınayan *isleap* isimli bir işlev yazın:

```
int is_leap(int year);
```

***isleap* işlevi kendisine gönderilen yıl bir artık yılsa sıfır dışı bir değere, artık yıl ise *0* değerine geri dönmeli. Yazdığınız kodu aşağıdaki *main* işleviyle sınayabilirsiniz:**

```
#include <stdio.h>

int isleap(int year);

int main()
{
    int year;
```

```
    printf("bir yıl giriniz: ");
    scanf("%d", &year);
    if (isleap(year))
        printf("%d yılı  artık yıl!..\n", year);
    else
        printf("%d yılı  artık yıl değil!..\n. year");

    return 0;
}
```

****

23. HANGİSİ BÜYÜK HANGİSİ KÜÇÜK?

Klavyeden alınan *int* türden üç sayı arasındaki büyüklük - küçüklük ilişkisini küçükten büyüğe doğru < ve = simgeleriyle gösterin. Program üç tane int türden sayı isteyerek, aralarındaki ilişkiyi ekranda göstermeli. İşte birkaç örnek:

```
Giriş  : 10 20 30
Yanıt : 10 < 20 < 30

Giriş  : 30 10 20
Yanıt : 10 < 20 < 30

Giriş  : 10 10 15
Yanıt : 10 = 10 < 15
```

****

24. AKREPLE YELKOVAN ARASINDAKİ AÇI

Kendisine gönderilen saat ve dakika değerlerini kullanarak akrep ile yelkovan arasındaki açıyı, "derece" cinsinden hesaplayarak geri dondüren "*get_angle*" isimli işlevi yazın. İşlevin bildirimi aşağıdaki gibidir:

```
double get_angle(int hour, int minute);
```

Yazdığınız kodu aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

double get_angle(int hour, int minute);


int main()
{
    int h, min;

    printf("saat ve dakika değerlerini giriniz :");
    scanf("%d%d", &h, &min);
    printf("açı = %lf\n", get_angle(h, min));

    return 0;
}
```

****

25. BASAMAK SAYISINI BULMAK

Kendisine gönderilen *int* türden bir sayının basamak sayısına geri dönen *no_of_digits* işlevini tanımlayın.

```
int no_of_digits(int value);
```

Yazmış olduğunuz işlev aşağıdaki kod ile sınayabilirsiniz:

```
#include <stdio.h>

int no_of_digits(int value);

int main()
{
    int number;
    int n = 5;

    while (n--> 0) {
        printf("bir sayı giriniz : ");
        scanf("%d", &number);
        printf("%d sayısı %d basamaklı\n", number, no_of_digits(number));
    }

    return 0;
}
```

****

26. BASAMAK DEĞERLERİNİN TOPLAMI

Kendisine gönderilen *int* türden bir sayının basamak değerleri toplamıyla geri dönen *sum_digits* işlevini tanımlayın:

```
int sum_digits(int value);
```

Yazmış olduğunuz işlev aşağıdaki kod ile sınayabilirsiniz:

```
#include <stdio.h>

int sum_digits(int value);

int main()
{
    int val;
    int n = 5;

    while (n-- > 0) {
        printf("bir sayı giriniz : ");
        scanf("%d", &val);
        printf("%d için basamaklar toplamı =  %d\n", val, sum_digits(val));
    }

    return 0;
}
```

****

27. ORTAK BASAMAKLAR

Kendisine gönderilen iki sayıdan, birincisinin rakamları ile ikinci sayının elde edilip edilemiyeceğini sınayan *is_possible* isimli işlevi yazın. İşlevin bildirimi:

```
int is_possible(int num1, int num2);
```

İşlev eğer yazmak mümkünse sıfır dışı bir değere, değilse *0* değerine geri dönmeli.

```
2735    5273    Yazılabilir.
2753    25333   Yazılabilir.
28      823     Yazılamaz
223     32      Yazılabilir
```

****

28. BÖLENLERİN TOPLAMI

Kendisine gönderilen pozitif bir tamsayının, kendisi hariç tüm çarpanlarının toplamı ile geri dönen *sum_factors* işlevini tanımlayın:

```
int sum_factors(int value);
```

***sum_factors* işlevi, *value* sayısının kendisi hariç tüm çarpanlarının toplamına geri dönmeli. *1* her sayının doğal çarpanı olduğundan, toplama eklenmeli.**
Yazdığınız kodu aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int sum_factors(int value);

int main()
{
    int k;

    for (k = 1200; k <= 1250; k++)
        printf("%d çarpanlari toplami = %d\n", k, sum_factors(k));

    return 0;
}
```

****

29. ARKADAŞ SAYILAR

x, y pozitif tamsayılar olmak üzere, eğer *x* sayısının çarpanları toplamı *y* sayısına, ve aynı zamanda *y* sayısının çarpanları toplamı x sayısına eşit ise, bu sayılar "arkadaştır" denir. Örneğin *220* ve *284* arkadaş sayılardır:

```
220 => 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284
284 => 1 + 2 + 4 + 71 + 142 = 220
```

Kendisine gönderilen iki tamsayının arkadaş olup olmadıklarını sınayan, *are_friends* işlevini tanımlayın:

```
int are_friends(int number1, int number2);
```

****

30. DÖRT BASAMAKLI ARKADAŞ SAYILAR

Dört basamaklı tüm arkadaş sayı çiftlerini bulan bir C programı yazın.
Not: Her bir çift bir kez yazılmalı.

****

31. NIVEN (HARSHED) SAYILARI

Eğer bir tamsayı basamaklarının toplamına tam olarak bölünüyorsa, bu tamsayı "Niven" (Harshed) tamsayısıdır. Verilen iki tamsayı aralığında kaç tane "Niven" sayısı olduğunu

hesaplayan bir program yazınız.Not : *1987000* ile *1988000* arasında *65 Niven* sayısı vardır.

****

32. NIVEN (HARSHED) ARKADAŞ SAYILARI

Bir arkadaş sayı çiftine konu tamsayıların ikisi de *niven*(*harshed*) tamsayısı ise, bu sayılara "harshed arkadaş sayıları" denir.
***2620* ve *2924 Harshed* arkadaş sayı çiftidir.**

***2620*, *2924* arkadaş sayılardır.**
***2620*, *10* tamsayısına tam olarak bölünür : 10 = 2 + 6 + 2 + 0**
2924, 17 tamsayısına tam olarak bölünür : 17 = 2 + 9 + 2 + 4
Başka *Harshed* Arkadaş sayı örnekleri:

```
10634085 ve 14084763
23389695 ve 25132545
34256222 ve 35997346
```

Verilen bir tamsayı aralığındaki "Harshed arkadaş sayıları" nı ekrana yazdıran bir program yazın.

****

33. SAYI ASAL MI?

Kendisine gönderilen pozitif bir tamsayının asal olup olmadığını sınayan, *isprime* işlevini tanımlayın.

```
int isprime(int number);
```

***isprime* işlevi kendisine gönderilen sayı asal ise sıfır dışı bir değere, asal değil ise *0* değerine geri dönmeli. *0* ve *1* sayıları asal değildir.**
Yazdığınız kodu aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int isprime(int number);

int main()
{
    int k;

    for (k = 0; k < 1000; ++k)
        if (isprime(k))
            printf("%d ", k);

    return 0;
}
```

****

34. EN YAKIN ASAL SAYI

Argüman olarak gönderilen *int* türden sayıdan küçük ilk asal sayıyı bulan *closest_prime* isimli işlevi yazın. İşlevin bildirimi

```
int closest_prime(int value);
```

İşlevin geri dönüş değeri, kendisine gönderilen değerden daha küçük olan en büyük asal sayı olmalı. Bu koşulu sağlayan bir asal sayı olmaması durumunda işlev *0* değerine geri dönmeli. Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int closest_prime(int value);

int main()
{
    int a;

    scanf("%d", &a);

    if (closest_prime(a))
        printf("%d'dan küçük olan en büyük asal sayı %d'dir\n", a,
closest_prime(a));
    else
        printf("%d'dan küçük bir asal sayı yoktur.");

    return 0;
}
```

****

35. n'İNCİ ASAL SAYI

***n* 1'den büyük ya da 1'e eşit bir tamsayı olmak üzere n. asal sayıyı bulan bir işlev tanımlayın.**

```
int nprime(int n);
```

İşlevin geri dönüş değeri *n*. asal sayı olmalı.

****

36. BASAMAKLARLA YAZILACAK TÜM ASAL SAYILAR

Kendisine gönderilen bir sayının basamakları ile yazılabilecek tüm asal sayıları, ekrana küçükten büyüğe doğru yazan bir işlev yazın. İşlevin bildirimi aşağıdaki gibidir:

```
void print_primes(int value);
```

Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

void print_primes(int value);

int main()
{
    print_primes(1729);

    return 0;
}
```

****

37. ASAL SAYI BULMACASI

***abc*, üç basamaklı bir tamsayıdır. Aşağıdaki koşulları sağlamaktadır:**

koşul 1 cba > abc olacak
koşul 2 abc asal sayı olacak
koşul 3 cba asal sayı olacak
koşul 4 ab asal sayı olacak
koşul 5 bc asal sayı olacak
koşul 6 cb asal sayı olacak
koşul 7 ba asal sayı olacak

113 sayısı bu koşulları sağlamaktadır:

koşul 1 cba > abc olacak 311 > 113
koşul 2 abc asal sayı olacak 113 asal
koşul 3 cba asal sayı olacak 311 asal
koşul 4 ab asal sayı olacak 11 asal
koşul 5 bc asal sayı olacak 13 asal
koşul 6 cb asal sayı olacak 31 asal
koşul 7 ba asal sayı olacak 11 asal

Bir C programı yazarak, yukarıdaki koşulların hepsini sağlayan tüm abc sayılarını bulun.

****

38. ASAL SAYI TOPLAMLARI

***2* basamaklı birbirinden farklı *3* asal sayının toplamı şeklinde ifade edilebilecek en büyük ve en küçük asal sayılar hangileridir?**
***3* basamaklı birbirinden farklı *3* asal sayının toplamı şeklinde ifade edilebilecek en büyük ve en küçük asal sayılar hangileridir?**
Bir C programı yazarak bu sayıları bulun.

****

39. ÜS HESAPLAMAK

Bir pozitif tamsayının başka bir pozitif kuvvetini hesaplayan, *power* isimli işlevi tanımlayın:

```
int power(unsigned int base, unsigned int exp);
```

power işlevi *base* ^{exp} ile geri dönmeli.
Yazmış olduğunuz işlev aşağıdaki kod ile sınayabilirsiniz:

```
#include <stdio.h>

int power(unsigned int base, unsigned int exp);

int main()
{
    int taban, us, k;

    for (k = 0; k < 10; ++k) {
        printf("taban ve us değerlerini giriniz  : ");
        scanf("%d%d", &taban, &us);
        printf("%d sayısının %d. kuvveti = %d\n", taban, us, power(taban,
us));
    }
```

```
    return 0;
}
```

****

40. BİNOM ÜÇGENİ

Klavyeden girilen bir "n" sayısı için binom üçgeninin ilgili satırındaki katsayıları ekrana yazan programı yazın:

"n" sayısı yukarıdaki gibi iki bilinmeyenli *n.* dereceden bir denklemin derecesini gösteriyor.

```
n = 0 icin :   1
n = 1 icin :   1   1
n = 2 icin :   1   2   1
n = 3 icin :   1   3   3   1
n = 4 icin :   1   4   6   4   1
n = 5 icin :   1   5   10   10   5   1
n = 6 icin :   1   6   15   20   15   6   1
```

Program girilen "n" degerine karşılık, binom üçgeninin ilgili satırını yukarıdaki gibi yazmalı.

****

41. TOMBALA OYUNU OLASILIĞI

Bir tombala torbasında *1*'den *99*'a kadar numaralanmış (*99* dahil) pullar bulunuyor. Bu tombala torbasıyla aşağıdaki oyunlar oynanıyor:

Çekilen bir pul torbaya geri atılmamak üzere

1) Torbadan 3 pul çekiliyor. Çekilen pulların toplamı 150'den küçük ise oyuncu kazanıyor.
2) Torbadan 3 pul çekiliyor. çekilen pulların toplamı asal sayı ise oyuncu kazanıyor.
3) Torbadan 3 pul çekiliyor. En büyük değerli pul ile en küçük değerli pul arasındaki fark ortanca değerli puldan büyükse oyuncu kazanıyor.

Oynanacak her bir oyun için oyuncunun kazanma olasılığını en az 30000 oyunu simule ederek hesaplayın.

****

42. CRAPS OYUNU

***Craps* hemen hemen dünyanın her yerinde bilinen, iki zarla oynanan bir kumardır.**
Oyunun kuralları şöyledir :
Zarları atacak oyuncu oyunu kasaya (kumarhaneye) karşı oynar. Atılan iki zarın toplam değeri

***7* ya da *11* ise oyuncu kazanır.**
***2, 3, 12* ise oyuncu kaybeder. (buna craps denir!)**

İki zarın toplam değeri yukarıdakilerin dışında bir değer ise (yani 4, 5, 6, 8, 9, 10) oyun şu şekilde sürer:
Oyuncu aynı sonucu buluncaya kadar zarları yeniden atar. Eğer aynı sonucu bulamadan önce oyuncu 7 atarsa (yani atılan iki zarın toplam değeri 7 olursa) oyuncu kaybeder.
Eğer 7 gelmeden önce oyuncu aynı sonucu yeniden atmayı başarırsa , kazanır.

Birkaç örnek :

Oyuncu zarları attı, zarların toplam değeri :
11 oyuncu kazandı. Yeni oyun oynanacak.
3 oyuncu kaybetti. Yeni oyun oynanacak.
12 oyuncu kaybetti. Yeni oyun oynanacak.
7 oyuncu kazandı. Yeni oyun oynanacak.

9 sonuç belli değil, oyuncu tekrar zar atacak.
8 sonuç belli değil, oyuncu tekrar zar atacak.
11 sonuç belli değil oyuncu tekrar zar atacak.
5 sonuç belli değil oyuncu tekrar zar atacak.
9 oyuncu kazandı. (7 atmadan aynı zarı tekrar attı). Yeni oyun oynanacak.

6 sonuç belli değil, oyuncu tekrar zar atacak.
3 sonuç belli değil, oyuncu tekrar zar atacak.
10 sonuç belli değil, oyuncu tekrar zar atacak.
7 oyuncu kaybetti. (Aynı zarı tekrar atamadan 7 geldi)

Böyle bir oyunu bir kez oynadığınızda kazanma olasılığınız nedir?
Bu oyunun benzetimini (simulasyon) yaparak bilgisayara 100.000 kez oynatan, ve oyuncunun kazanma olasılığını hesaplayan bir C programı yazınız.

****

43. e, n, d HARFLERİNİ GİRİNCE SONLANAN PROGRAM

Klavyeden alınan her karakteri ekrandan gösteren, ancak klavyeden ardışık olarak 'e', 'n', 'd' karakterleri alındığında sonlanan bir C programı yazın.

****

44. e SAYISININ HESAPLANMASI

***e* sayısını aşağıdaki seri toplamıyla bulabilirsiniz:**

```
e = 1/0! + 1/1! + 1/2! + 1/3! + 1/4! + 1/5!
```

Bir C programı yazarak *e* sayısını yukarıdaki seri toplamıyla hesaplayarak ekrana yazdırın.

****

45. EKRANA RASTGELE SÖZCÜKLERİN YAZDIRILMASI

Aşağıdaki koşulları sağlayan bir C programı yazın:
Ekrana her satıra bir tane olmak üzere *NO_OF_WORDS* tane rastgele sözcük yazdırılmalı.
Her sözcük İngiliz alfabesindeki büyük harf karakterlerinden oluşturulmalı.
Her bir sözcüğün uzunluğu en az *MIN_WORD_LEN* en fazla *MAX_WORD_LEN* olmalı.
Bir sözcüğün içinde *NO_SUC_VOWELS* sayıda sesli harf yanyana gelmemeli.
Bir sözcüğün içinde *NO_SUC_CONS* sayıda sesli harf yanyana gelmemeli.

Programınızı aşağıdaki "simgesel değişmez" değerleriyle çalıştırın:

```
#define  NO_OF_WORDS       10
#define  MIN_WORD_LEN      9
#define  MAX_WORD_LEN      15
#define  NO_SUC_VOWELS     2
#define  NO_SUC_CONS       3
```

Aşağıda örnek bir ekran çıktısı görülüyor.

```
HQUMEYLIRCUXWOZ
SREPGEVIKEFFI
KKASVENZADPOF
ZBOJUVPABOY
POYLIPYAMYE
WQUJJOVAWUXWOK
FZATXIXJUF
UXXOKETLOGZ
RCITEJWAYYAJL
VGUBFAVLES
```

****

47. FAKTÖRİYEL HESAPLAYAN İŞLEV

Kendisine gönderilen bir tamsayıya ilişkin faktöriyel değeriyle geri dönen *factorial* işlevini tanımlayın:

```
int factorial(int);
```

Yazmış olduğunuz işlevi aşağıdaki main işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int factorial(unsigned int);

int main()
{
    int k;

    for (k = 0; k < 14; ++k)
        printf("%d! = %ld\n", k, factorial(k));

    return 0;
}
```

****

48. HAFTANIN HANGİ GÜNÜ?

Gün, ay ve yıl olarak verilen geçerli bir tarihin haftanın hangi gününe geldiğini hesaplayan

```
int day_of_week(int day, int month, int year);
```

işlevini tanımlayın. İşlev 01.01.1900 tarihinden sonraki tüm tarihler için doğru çalışmalı. İşlev pazar günü için *0* değerine, Salı günü için 1 değerine... Cumartesi günü için *6* değerine geri dönmeli. Tanımladığınız işlevin doğru çalışıp çalışmadığını görmek için çeşitli tarih değerleriyle sınayın.

****

49. KAPI VE ANAHTAR BULMACASI

***100* tane kapı var. Bu kapılar *1*'den *100*'e kadar numaralandırılmış.**
Kapıların hepsi kilitli.
Elimizde *1*'den *20*'ye kadar numaralandırılmış 20 tane anahtar var.

Bir kapının numarası bir anahtarın numarasına tam bölünüyorsa o anahtar o kapıyı açabiliyor (ya da kilitleyebiliyor!)
1 **numaralı anahtardan başlayarak her anahtar için kapılar dolaşılıyor. Anahtar ile kapı açıksa kilitleniyor, kilitliyse açılıyor.**
20. turdan sonra, yani son anahtarın kullanılmasından sonra hangi kapılar açık hangi kapılar kilitlidir? Bir C programı yazarak açık olan kapıların numaralarını ekrana yazdırın.

****

50. KLAVYEDEN GİRİLEN SAYILARIN ORTALAMASINI BULMAK

Klavyeden girilen tamsayı değerlerin, toplam sayısını, en büyüğünü, en küçüğünü, ortalamasını hesaplayan "deger.c" isimli bir C programı yazın. Kullanıcının her değer girişinden önce, yeni bir değer girmek isteyip istemediği sorulmalı:

```
Yeni bir deger girmek istiyor musunuz? [e] [h]
```

Kullanıcı 'e' ya da 'e' tuşuna basarsa, program kullanıcının bir değer girmesini istemeli:

```
bir tamsayı giriniz: 53 enter
```

Kullanıcı 'h' ya da 'h' tuşuna basarsa, program o ana kadar girilen değerlerle ilgili olarak aşağıdaki dökümü olarak vererek sonlanmalı:

```
toplam 17 deger girildi.
max = 89
min = 17
ortalama = 37.456786
program sonlandı!
```

Kullanıcının 'e', 'e', 'h' ya da 'h' tuşları dışında başka bir tuşa basması durumunda program bir tepki vermemeli.
Programda dizi kullanılmamalı.
Girilen değerler [0 - 100] aralığında olmalı. Geçersiz bir değer girilmesi sırasında ekrana

```
"geçersiz değer "
```

uyarısı yazılarak yeni bir giriş yapılması istenmeli.

```
"yeni bir deger giriniz"
```

****

51. MÜKEMMEL SAYI SINAMASI

Kendisi haricindeki tüm çarpanlarının toplamına eşit olan tamsayılara "mükemmel sayı" (*perfect number*) denir. Örneğin *6* ve *28* mükemmel sayılardır:

```
6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14
```

Kendisine gönderilen int türden bir değerin mükemmel sayı olup olmadığını sınayan *is_perfect* işlevini tanımlayın:

```
int is_perfect(int value);
```

is_perfect **işlevi kendisine gönderilen tamsayı mükemmel ise sıfır dışı bir değere, mükermmel değil ise *0* değerine geri dönmeli.**
****

52. 4 BASAMAKLI MÜKEMMEL SAYI

_4_ basamaklı tek bir mükemmel sayı vardır. Yazmış olduğunuz _isperfect_ işlevini kullanarak 4 basamaklı mükemmel sayıyı bulan bir C programı yazın.

****

53. OLASILIK SİMULASYONU

Yazı tura ile oynanan aşağıdaki oyunu kazanma olasılığınız nedir?
Bir C programı ile _1.000.000_ oyunun benzetimini yaparak bulun.

- Oyun sizinle _x_ arasında geçiyor. İkinizin de _100_ YTL'si var.

- Paranın her atılmasından önce x size _10_ YTL veriyor.

- Üst üste iki yazı geldiğinde, siz ona _35_ YTL, üst üste üç tura geldiğinde siz ona _60_ YTL veriyorsunuz.

Bir tarafın parası bitince oyun sona eriyor.

****

54. ORTAK BÖLENLERİN EN BÜYÜĞÜNÜ BULMAK

İki tamsayının ortak bölenlerinin en büyüğünü hesaplayan _obeb_ işlevini yazın:

```
int obeb(int sayi1, int sayi2);
```

İşlev, parametre değişkenlerine geçilen tamsayıların ortak bölenlerinin en büyüğü değeriyle geri dönmeli.
Yazdığınız işlevi aşağıdaki _main_ işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int obeb(int sayi1, int sayi2);

int main()
{
    int s1, s2;

    printf("iki sayı giriniz : ");
    scanf("%d%d", &s1, &s2);
    printf("%d ve %d sayılarının o.b.e.b'i  = %d\n", s1, s2, obeb(s1, s2));

    return 0;
}
```

****

55. ORTAK KATLARIN EN KÜÇÜĞÜ

İki tamsayının ortak katlarının en küçüğünü hesaplayan _okek_ işlevini yazın:

```
int okek(int sayi1, int sayi2);
```

İşlev parametre değişkenlerine geçilen tamsayıların ortak katlarının en küçüğü değeriyle geri dönmeli.
Yazdığınız işlevi aşağıdaki _main_ işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int okek(int sayi1, int sayi2);

int main()
{
    int s1, s2;

    printf("iki sayı giriniz : ");
    scanf(“%d%d", &s1, &s2);
    printf("%d ve %d sayılarının o.k.e.k'i = %d\n", s1, s2, okek(s1, s2));

    return 0;
}
```

****

56. OYUN SİMULASYONU

Bir C programı yazarak aşağıdaki oyunda *A* oyuncusunun kazanma olasılığını hesaplayın:

1. Oyun *A* ile *B* arasında oynanır. Oyuna başlangıçta *A*'nın ve B'nin *100* YTL parası vardır. A, her bir oyuna girmek için B'ye 5 YTL verir. Yeni bir oyuna girmek için A'nın eğer *5 YTL* parası yoksa A oyunu kaybeder.
B, 0 - ile 9 arasında (0 ve 9 dahil olmak üzere) bir sayı tutar. A, bu sayıyı 3 tahminde bulmaya çalışır.

1. Eğer A, *B*'nin tuttuğu sayıyı ilk tahminde bulursa *B*, A ya *20* YTL öder.
2. Eğer A, *B*'nin tuttuğu sayıyı ikinci tahminde bulursa *B*, *A* ya *15 YTL* öder.
2. Eğer A, *B*'nin tuttuğu sayıyı üçüncü tahminde bulursa B A ya 10 YTL öder.

***B*'nin tuttuğu sayıyı A nın bulması durumunda *B*'de ödeme yapacak kadar para bulunmamsı durumunda B oyunu kaybeder.**
Bu oyunu A nın kazanma olasılığı nedir?

****

57. Pİ SAYISINI HESAPLAMAK

***pi* sayısını aşağıdaki seri toplamıyla bulabilirsiniz:**

```
pi / 4 = 1/1 - 1/3 + 1/5 - 1/7 + 1/9 ....
```

***pi* sayısını yukarıdaki seri toplamıyla hesaplayarak ekrana yazdıran bir C programı yazın.**

****

58. PRIMEX SAYILAR

Bildirimi aşağıda verilen *isprimex* işlevini tanımlayın.

```
int isprimex(int number);
```

***isprimex,* kendisine gönderilen argümanın asal olup olmadığını sınamalı. Eğer sayı asal ise bu kez sayının basamak değerleri toplanarak elde edilen sayının asal olup olmadığı sınanmalı. Bu işlem sonuçta tek basamaklı bir sayı kalana kadar sürmeli. Eğer en son elde edilen tek basamaklı sayı dahil tüm sayılar asal ise *isprimex* işlevi sıfır dışı bir değere geri dönmeli. Eğer herhangi bir kademede asal olmayan bir sayı elde edilirse işlev, 0 değerine geri dönmeli.**
Not : Birden fazla işlev tanımlayabilirsiniz.

Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int isprimex(int number);

int main()
{
    int k;

    for (k = 19000; k <= 19500; ++k)
        if (isprimex(k))
            printf("%d \n", k);

    return 0;
}
```

****

59. RASTGELE BİR TARİH YAZDIRMAK

Çağrıldığında ekrana *01.01.1900* ile *31.12.2000* arasında rastgele bir tarih yazacak

```
void print_random_date(void);
```

işlevini tanımlayın. İşlevin ekrana yazdırdığı tarih geçerli bir tarih olmalı. Rastgele tarihin *Şubat* ayına denk gelmesi durumunda, seçilen yılın artık yıl olup olmamasına göre, gün değeri *29* olabilmeli. Tarih ekrana aşağıdaki formatta yazdırılmalı:

```
03rd Jan 1857
```

Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

void print_random_date(void);

int main()
{
    int k;

    for (k = 0; k < 20; ++k) {
        print_random_date();
        putchar('\n');
    }
    return  0;
}
```

****

60. HER SATIRA DAHA FAZLA YILDIZ.

Aşağıda bildirimi verilen *put_stars* isimli işlevi tanımlayın:

```
void put_stars(int n);
```

İşlev çağırıldığı zaman n değeri kadar satıra '*' karakteri basmalı. Birinci satıra bir '*' karakteri, ikinci satıra iki '*' karakteri, üçüncü satıra üç '*' karakteri... *n*.satıra *n* adet '*' karakteri.:

```
*
**
***
****
*****
```

Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

void put_stars(int val);

int main()
{
    int n;

    printf("bir sayı girin : ");
    scanf("%d", &n);
    put_stars(n);

    return 0;
}
```

****

61. MASTERMİND OYUNU

***Mastermind* iki kişi ile oynanan bir sayı bulmaca oyunudur. Oyunculardan biri, [*1000 - 9999*] kümesinden basamak değerleri birbirinden farklı bir sayı belirler. Diğer oyuncunun amacı bu sayıyı en fazla *10* tahminde bulmaktır. Sayıyı belirleyen oyuncu, diğer oyuncunun her tahmininden sonra oyunun kuralları doğrultusunda bilgiler verir.**
Tahmin edilen sayı içinde belirlenen sayının içerisindeki rakamlardan biri varsa, fakat basamak değeri tutmuyorsa - işareti ile, basamak değeride tutuyorsa + işareti ile bilgi verilir. Örnekler:

```
Belirlenen sayı        : 1234
Tahmin edilen sayı     : 4567
Verilecek bilgi        : -

Belirlenen sayı        : 1234
Tahmin edilen sayı     : 5674
Verilecek bilgi        : +

Belirlenen sayı        : 1234
Tahmin edilen sayı     : 4237
Verilecek bilgi        : +2  -
```

Not : verilecek olan bilgide + ve - 'lerin sırasının bir önemi yoktur.

Bir sayı belirleyerek *mastermind* oyununu oynatan programı yazın.

****

62. SMITH SAYILARI

***1*'den büyük asal olmayan bir tamsayının rakamlarının toplamı, sayı asal çarpanlarına ayrılarak yazıldığında bu yazılışta bulunan tüm asal sayıların rakamlarının toplamına eşit oluyorsa bu tür sayılara "Smith sayısı" denir.**
Örneğin:

```
728 = 2 * 2 * 2 * 7 * 13
7 + 2 + 8 = 2 + 2 + 2 + 7 + 1 + 3
```

olduğundan *728* bir *Smith* sayısıdır.
***1* ile *10000* arasındaki tüm *Smith* sayılarını bularak ekrana yazdıran bir C programı yazın.**

****

63. TÜRKÇE KARAKTERİ DÖNÜŞTÜREN İŞLEV

Kendisine gönderilen karakter türkçe'nin özel karakterlerinden biri ise (Ç, Ğ, İ, Ö, Ş, Ü, Ç, ğ, ı, ö, ş, ü) bu karakterin ingilizce benzerininin sıra numarasına geri dönen, aksi halde gönderilen karakterin sıra numarasına geri dönen *to_eng* işlevini yazın:

```
int to_eng(int ch);
```

İşlevin tasarımında *switch* kontrol deyimini kullanın. Otomatik tür dönüşümü nedeniyle oluşabilecek böceklere karşı önlem alın!
Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

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

#define    ESC    0x001B

int to_eng(int ch);

int main()
{
    int ch;

    while ((ch = getch()) != ESC)
        putchar(to_eng(ch));

    return 0;
}
```

****

64. TOMBALA VE OLASILIK BENZETİMİ

Bir tombala torbasında *1*'den *99*'a kadar numaralanmış (99 dahil) pullar bulunuyor. Bu tombala torbasıyla aşağıdaki oyunlar oynanıyor:
Çekilen bir pul torbaya geri atılmamak üzere

1) Torbadan 3 pul çekiliyor. Çekilen pulların toplamı 150'den küçük ise oyuncu kazanıyor.
2) Torbadan 3 pul çekiliyor. Çekilen pulların toplamı asal sayı ise oyuncu kazanıyor.
3) Torbadan 3 pul çekiliyor. En büyük değerli pul ile en küçük değerli pul arasındaki fark ortanca değerli puldan büyükse oyuncu kazanıyor.

Oynanacak her bir oyun için oyuncunun kazanma olasılığını en az *30000* oyunun benzetimini (simulasyon) yaparak hesaplayın.

****

65. urand İŞLEVİ

Çağrıldığında *0* ile LIMIT değerleri arasında rastgele farklı bir tamsayı üreten *u_rand* isimli işlevi tanımlayın:

```
int urand(void);
```

İşlev verilen aralıktaki tüm rastgele değerler üretildikten sonra çağrılırsa, -1 değerine geri dönmeli. Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

#define LIMIT   100

int urand(void);

int main()
{
    int k;

    for (k = 0; k <= LIMIT; ++k)
        printf("%d ", urand());
    return 0;
}
```

****

66. YILIN KAÇINCI GÜNÜ?

Kendisine gönderilen gün, ay ve yıl değerlerinden oluşan bilgiyi bir tarih olarak ele alıp söz konusu tarihin yılın kaçıncı günü olduğu bilgisiyele geri dönen *day_of_year* işlevini tanımlayın:

```
int day_of_year(int day, int mon, int year);
```

Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

int day_of_year(int day, int mon, int year);

int main()
{
    int d, m, y;

    printf("gun ay ve yıl değerlerini giriniz: ");
    scanf("%d%d%d", &d, &m, &y);
    printf("%d yılının %d. günü\n", day_of_year(d, m, y));

    return 0;
}
```

****

67. İKİNCİ DERECEDEN BİR DENKLEMİ ÇÖZMEK

İkinci dereceden bir denklemi çözen bir C programı yazın. İkinci dereceden bir denklem

```
ax2 + bx + c = 0
```

biçimindedir. *a*, *b*, ve *c* gerçek sayılardır. Program kullanıcıdan *a*, *b* ve *c* katsayılarını almalı. Programın çıktısı aşağıdaki biçimlerde olabilir:

```
denkleminizin gerçek kökü yoktur!
```

veya

```
denkleminizin tek gerçek kökü vardır:
kok = 3.750000
```

veya

```
denkleminizin iki gerçek kökü vardır:
kok1 = -3.250000
kok2 = 2.000000
```

****

68. BİR SAYIYI İKİLİK SAYI SİSTEMİNDE YAZDIRMAK

Kendisine gönderilen *int* türden bir tamsayıyı ekrana ikilik sayı sisteminde yazdıracak

```
void print_binary(int val);
```

işlevini tanımlayın. Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

void print_binary(int val);

int main()
{
    int k;

    for (k = 17800; k <= 17820; ++k) {
        printf("%d = ");
        print_binary(k);
        putchar('\n');
    }
    return 0;
}
```

****

69. TAMSAYIYI TERS ÇEVİREN İŞLEV

Kendisine gönderilen int türden bir değerin tersiyle (basamaklarının ters çevrilmiş biçimiyle) geri dönen *reverse_val* isimli işlevi tanımlayın. İşlevin bildirimi

```
int reverse_val(int value);
```

biçimindedir.
***reverse_val* işlevi parametre değişkenine kopyalanan değerin basamakları ters çevrilerek elde edilen değere geri dönmeli. Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:**

```
#include <stdio.h>

int reverse_val(int value);

int main()
{
    int number;

    printf("bir sayı giriniz : ");
```

```
    scanf("%d", &number);
    printf("%d sayısının tersi = %d\n", number, reverse_val(number));

    return 0;
}
```

****

70. KLAVYEDEN GİRİLEN SAYILARIN ORTALAMASINI BULMAK

Klavyeden girilen tamsayı değerlerin, toplam sayısını, en büyüğünü, en küçüğünü, ve ortalamasını hesaplayan *deger.c* isimli bir C programı yazın. Kullanıcının her deger girişinden önce, yeni bir değer girmek isteyip istemediği sorulmalı:

```
Yeni bir deger girmek istiyor musunuz? [E] [H]
```

Kullanıcı *'e'* ya da *'e'* tuşuna basarsa, program kullanıcının bir değer girmesini istemeli.

```
Bir tamsayı giriniz: 53 enter
```

Kullanıcı *'h'* ya da *'H'* tuşuna basarsa, program o ana kadar girilen değerlerle ilgili olarak aşağıdaki dökümü olarak vererek sonlanmalı:

```
Toplam 17 deger girildi.
Max = 89
Min = 17
Ortalama = 37.456786
program sonlandı!
```

Kullanıcının '*E*', '*e*', '*H*' ya da '*h*' tuşları dışında başka bir tuşa basması durumunda program bir tepki vermemeli.

1. Programda dizi kullanılmayacak.
2. Girilen değerler [0 - 100] aralığında olmalı. Geçersiz bir değer girilmesi sırasında ekrana

```
"geçersiz değer "
```

uyarısı yazılarak yeni bir giriş yapılması istenmeli.

```
"yeni bir deger giriniz"
```

****

71. DİZİNİN EN BÜYÜK ELEMANINI BULMAK

***int* türden bir dizinin en büyük elemanını bulan bir kod parçası yazın:**

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

#define    SIZE    40

int main()
{
    int a[SIZE];
    int k, max;
```

```
    srand(time(0));

    for (k = 0; k < SIZE; ++k) {
        a[k] = rand() % 500;
        if (k && k % 20 == 0)
            printf("\n");
        printf("%3d ", a[k]);
    }
    printf("\n");

    /* KOD */

    return 0;
}
```

****

72. DİZİNİN EN BÜYÜK İKİNCİ ELEMANINI BULMAK

int **türden bir dizinin en büyük ikinci elemanını bulan bir kod parçası yazın. Bir önceki soru için verilen sınama kodunu kullanabilirsiniz.**

****

73. DİZİNİN STANDART SAPMASINI BULMAK

Standart sapma ortalamadan sapmanın bir ölçüsüdür. Standart sapmayı aşağıdaki formül ile hesaplayabilirsiniz :

ort = dizinin aritmetik ortalaması
n = dizinin eleman sayısı
a[i] = dizinin her bir elemanı
farkkare[i] = (ai - ort) * (ai - ort)
standart sapma = karekok (kumulatif(farkkare[i]) / n)

Yukarıdaki formülde *ort* dizinin aritmetik ortalamasını, *n* ise dizinin eleman sayısını gösteriyor. "Bir dizinin standart sapması dizi elemanlarının ortalamadan farklarının kareleri kümülatif toplamınının dizinin eleman sayısına bölümünün karekköküne eşittir." *SIZE* elemanlı bir dizinin standart sapmasını *main* işlevi içinde hesaplayın.

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

#define SIZE    20

int main()
{
    int a[SIZE] = {1, 2, 6, 5, 4, 3, 7, 2,-2, 3, 4, 12,
                   31, -12, 15, 6, 0, 8, 19, 2};

    /*  standart sapma = 8.482924  */
}
```

****

74. DİZİNİN TEK VE ÇİFT ELEMANLARINI AYRI AYRI SIRALAMAK

SIZE **sayıda elemanı olan *int* türden bir dizinin elemanlarını, önce tek sayıları küçükten büyüğe, sonra çift sayıları küçükten büyüğe sıralayan, daha sonra dizi elemanlarını yazdıran bir C programı yazın. Örnek:**

```
int a[SIZE] = {3, 8, 12, 30, 56, 35, 78, 31, 69, 40};
```

dizisi verildiğinde ekrana dizi elemanları aşağıdaki şekilde yazdırılmalı:

```
a[0] = 3
a[1] = 31
a[2] = 35
a[3] = 69
a[4] = 8
a[5] = 12
a[6] = 30
a[7] = 40
a[8] = 56
a[9] = 78
```

****

75. DİZİNİN TEK VE ÇİFT ELEMANLARININ AYRI AYRI ORTALAMALARINI BULMAK

***SIZE* sayıda elemana sahip *int* türden bir dizinin tek ve çift elemanlarının aritmetik ortalamasını ayrı ayrı hesaplayan bir C programı yazın. Örnek:**

```
int a[SIZE] = {3, 8, 12, 30, 56, 35, 78, 31, 69, 40};
```

dizisi verildiğinde ekrana dizi elemanları aşağıdaki şekilde yazdırmalı:

```
tek sayıların ortalaması = 34.500000
çift sayıların ortalaması = 37.333333
```

Not: Dizide hiç tek sayı (ya da çift sayı) bulunmayabilir. Bu durumda sıfıra bölme hatası yapılmamalı.

****

76. SAYISAL LOTO KUPONU BASAN PROGRAM

Rastgele değerlerle *8* kolon sayısal loto oyunu oynayan bir C programı yazın:
*** bir kolonda [1 - 49] aralığında 6 sayı bulunmalı.**
*** bir kolonda aynı sayı iki kez bulunmamalı.**
*** iki kolon birbirinin aynı olabilir.**
*** kolonlar ekrana sıralı yazdırılmalı.**
Örnek bir program çıktısı :

```
KOLON 1 3   12  34  37  41  43
KOLON 2 14  19  20  30  40  45
KOLON 3 2   4   8   20  26  40
KOLON 4 12  17  19  25  28  34
KOLON 5 1   5   20  24  29  49
KOLON 6 12  23  28  30  34  37
KOLON 7 4   11  12  18  31  42
KOLON 8 9   18  27  29  38  44
```

****

77. İNDİS SIRALAMASI ALGORİTMASINI KODLAMAK

***SIZE* elemanlı bir *int* türden diziyi ilkdeğer vererek tanımlayın. Bu dizinin elemanlarını değil de elemanlarının indislerini sıraya dizerek başka bir dizi içerisine yerleştirin. Bu**

algoritmaya "indis sıralaması" denir. *SIZE* simgesel değişmezine istediğiniz değeri verebilirsiniz.

```
#include <stdio.h>

#define SIZE       5

int main()
{
    int a[SIZE] = {7, 9, 3, 51, 8};
    int b[SIZE];
    int i;

    /*
        YAZILACAK KOD
    */

    for (i = 0; i < SIZE; ++i)        /* 3, 1, 4,  0, 2 */
        printf("%d\n", b[i]);

    for (i = 0; i < SIZE; ++i)        /* 51, 9, 8, 7, 3 */
        printf("%d\n", a[b[i]]);

    return 0;
}
```

****

78. DİZİDE EN ÇOK YİNELENEN DEĞERİ BULMAK

***100* elemanlı *int* türden bir dizi içindeki en çok yinelenen değeri bulun. En çok yinelenen sayı birden fazla ise, bunlardan dizi içinde ilk görülen bulunmalı. Denemeyi diziye ilkdeğer vererek yapın:**

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

int main()
{
    int random[100];
    int k, mod;
    srand(time(0));

    for (k = 0; k < 100; ++k)
        random[k] = rand() % 50;
    /*  kod  */
    printf(" mod = %d\n", mod);
    for (k = 0; k < 100; ++k)
        printf("%d ", random[k]);

    return 0;
}
```

****

79. HER SATIRA BİR KARAKTER FAZLA

***char* türden bir dizi içinde bir yazı tutuluyor. Yazının ilk karakterini birinci satıra, ilk iki karakterini ikinci satıra, ilk üç karakterini üçücü.satıra... yazdıran bir C programı yazın. Diziye girilen yazının "*necati*" olduğunu düşünürsek aşağıdaki gibi olmalı:**

```
n
ne
nec
neca
necat
necati
```

****

80. YAZIDA HANGİ HARFTEN KAC TANE VAR?

Bir yazının içinde bulunan ingilizce harf karakterlerinden her birinin adedini ekrana listeleyin. Küçük büyük harf ayrımı yapılmamalı.

```
#include <stdio.h>

#define SIZE        100

int main()
{
    char str[SIZE];
    printf("bir yazi giriniz : ");
    gets(str);     /* “Necati Ergin” girildiğini düşünelim!"

    /* YAZILACAK KOD */

    return 0;
}
```

****

81. YAZIDA KAÇ SÖZCÜK VAR?

***char* türden bir dizi içinde tutulan bir yazıdaki sözcük sayısını hesaplayın. Yazının başında, içinde ya da sonunda boşluk karakterleri olabilir. Sınama amacıyla aşağıdaki kodu kullanın:**

```
#include <stdio.h>

#define    SIZE    200

int main()
{
    char str[SIZE];
    int word_counter = 0;
    /*  istediğiniz kadar değişken tanımlayabilirsiniz: */
    printf("bir yazı giriniz : ");
    gets(str);
    /*      YAZILACAK KOD  */
    printf("%s yazısı %d kelime\n", str, word_counter);

    return 0;
}
```

****

82. YAZIDAN TAMSAYI DEĞERİ ELDE ETMEK

***char* türden bir dizi içinde yazı olarak tutulan bir tamsayı değeri o dizinin içinden alarak bir değişkene atamaya çalışın:**

```
#include <stdio.h>

#define    SIZE    100

int main()
{
    char str[SIZE];
    int x = 0;

    printf("bir sayı giriniz : ");/* 5876 girildiğini düşünelim */
    gets(str);

    /*  YAZILACAK KOD  */

    printf("x = %d\n", x);     /* x = 5876 yazmalı */

    return 0;
}
```

****

83. YAZIDAN BAŞKA BİR YAZIDA OLAN KARAKTERLERİ SİLMEK

***char* türden bir dizi içinde tutulan bir yazının içinden, *char* türden başka bir dizi içinde tutulan yazıda bulunan tüm karakterleri silmeye çalışın:**

```
#include <stdio.h>

#define    SIZE    100

int main()
{
    char s1[SIZE] = "necati ergin";
    char s2[SIZE] = "nuri yılmaz";

    /*  YAZILACAK KOD */

    printf("yazı = %s", s1);   /* ecteg yazmalı" */

    return 0;
}
```

****

84. YAZININ SÖZCÜKLERİNİ TERSTEN YAZDIRMAK

***char* türden bir dizinin içindeki yazının sözcüklerini sondan başa doğru yazdırmaya çalışın:**

```
#include <stdio.h>

#define SIZE       100

int main()
{
    char str[100] = "iyi bir C programcısı olmak için çok çalışmak
gerekir";

    /* yazdığınız kodun çalışmasından sonra ekrana
         "çalışmak cok için olmak programcısı C bir iyi"
        yazmalı.
    */
```

```
    return 0;
}
```

****

85. YAZIYI AYRAÇ İÇİNE ALMAK

***char* türden bir dizi içinde tutulan yazıyı ayraç içine alan bir kod parçası yazın:**

```
# include <stdio.h>

#define SIZE       100

int main()
{
    char str[SIZE];

    printf("bir yazı giriniz : ");
    gets(str);                /* Necati Ergin */

    /*
        YAZILACAK KOD
    */
    puts(str);                /* (Necati Ergin) */

    return 0;
}
```

****

86. GÖSTERİCİ ALIŞTIRMALARI (1)

Aşağıdaki programda *printf* çağrısının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

```
#include <stdio.h>

int main()
{
    int a[] = {1, 2, 3, 4, 5};
    int *ptr = a;
    int k;

    *ptr++ = 10;
    *++ptr = 20;
    ++*ptr;
    ptr[2] = 30;
    *ptr++ = a[2];
    ptr[-2] = 40;

    for (k = 0; k < 5; ++k)
        printf("%d ", a[k]);

    printf("\n");

    return 0;
}
```

****

87. ANAHTAR SÖZCÜK MÜ?

Başlangıç adresini aldığı yazının standart *ANSI C* dilinin *32* anahtar sözcüğünden biri olup olmadığını sınayan

```
int is_keyword(const char *str);
```

işlevini tanımlayın. *str* adresindeki yazı geçerli bir anahtar sözcük ise işlev sıfır dışı bir değere, aksi halde *0* değerine geri dönmeli.

****

88. DİZİYİ TERS ÇEVİREN İŞLEV

Adresini ve boyutunu aldığı *int* türden bir diziyi ters çeviren, *reverse_array* isimli işlevi yazın:

```
int *reverse_array(int *ptr, int size);
```

İşlevin geri dönüş değeri ters çevrilen dizinin başlangıç adresi olmalı. Aşağıdaki sınama kodunu kullanabilirsiniz:

```
#include <stdio.h>

#define    SIZE   10

int main()
{
    int a[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    reverse_array(a, SIZE);

    for (k = 0; k < SIZE; ++k)
        printf("%d ", a[k]); /* 10 9 8 7 6 5 4 3 2 1 */

    return 0;
}
```

****

89. BÜYÜK TAMSAYILARLA İŞLEMLER

***200* basamağa kadar uzun tamsayılarla toplama, çıkarma ve çarpma işlemlerini yapan aşağıdaki işlevleri yazın. Sayılar *ASCII* karakterleri biçiminde karakter dizileri içinde saklanmaktadır.**

```
void addlong(const char *num1, const char *num2, char *num3);
void sublong(const char *num1, const char *num2, char *num3);
```

İşlevlerin ilk parametresi işleme sokulacak ilk sayıyı, ikinci parametresi ikinci sayıyı belirtmektedir. Sonuç üçüncü parametreyle belirtilen adrese yerleştirilmeli. Yerleştirme işlemininin sonunda '\0' karakteri eklemeyi unutmayın. Sayılar işaretli olabilir. Ancak başta ya da sonda herhangi bir boşluk karakteri içeremezler. Sonuç pozitifse sayının başına + simgesi eklenmemeli. Ancak negatifse – simgesi eklenmeli.
Aşağıdaki sayılar geçersizdir:

" - 9999"	**Başında boşluk var.**
"-9999 "	**Sonunda boşluk var.**
"99292 33"	**Ortasında boşluk var.**
"-12a"	**Sayısal olmayan bir karakter var.**

Ancak şu sayılar geçerlidir:

"00000123" **Sayı sıfırla başlayabilir**
"+12345" **sayının başına + getirilebilir.**

```
#include <stdio.h>

#define SIZE       (200 + 1)

int main()
{
    char n1[SIZE] =  "99999999999999999999999";
    char n1[SIZE] = "-99999999999999999988888888888888888";
    char n3[SIZE];

    addlong(n1, n2, n3);
    puts(n3);
    sublong(n1, n2, n3);
    puts(n3);

    return 0;
}
```

****

90. DİZİ SIRALI MI?

***int* türden bir dizinin küçükten büyüğe göre sıralı olup olmadığını sınayan *is_sorted* isimli işlevi yazın:**

```
int is_sorted(const int *ptr, int size);
```

***is_sorted* işlevi kendisine başlangıç adresi ile boyutu geçilen dizinin küçükten büyüğe doğru sıralanmış olup olmadığını sınamalı. Eğer dizi sıralı ise işlev sıfır dışı bir değere geri döner. Eğer dizi sıralı değil ise işlevin geri dönüş değeri *0* değeridir.**

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

#define SIZE   5

int is_sorted(const int *ptr, int size);

int main()
{
    int a[SIZE],  i, k;

    srand(time(0));
    for (i = 0; i < 10; ++i) {
        for (k = 0; k < SIZE; ++k) {
            a[k] =rand() % 5;
            printf( "%d ", a[k]);
        }
        printf("\n");

        if (is_sorted(a, SIZE))
            printf("dizi sıralı!\n");
        else
            printf("dizi sırasız!\n");
    }
    return 0;
}
```

****

91. GEÇERLİ BİR İSİM Mİ?

Kendisine gönderilen bir yazı içindeki degisken isminin C'de geçerli bir degişken ismi olup olmadığını sınayan *is_legal* isimli işlevi yazın. İşlev, aldığı isim geçerli bir değişken ismi ise sıfır dışı bir değere, geçerli bir değişken ismi değilse *0* değerine geri dönmeli.

```
int is_legal(const char var *str);
```

****

78. HECELERİ YAZDIRAN İŞLEV

Türkçe'nin heceleme kurallarına uygun olarak, verilen bir sözcüğü hece hece ekrana yazdıracak, aşağıda bildirimi verilen *hece_yaz* isimli işlevi tanımlayın:

```
int hece_yaz(const char *str)
```

***hece_yaz* işlevi başlangıç adresini aldığı yazıyı hece hece ekrana yazar. Hecelerin arasında yalnızca tek bir boşluk karakteri bulunmalı.**
***hece_yaz* işlevinin geri dönüş değeri ekrana yazılan hece sayısıdır.**

```
#include <stdio.h>

#define    SIZE 100

int hece_yaz(const char *str);

int main()
{
    char s[SIZE];

    printf("bir kelime giriniz : ");
    gets(s);
    printf("kelime hecelerine ayriliyor :\n");
    hece_yaz(s);

    return 0;
}
```

****

92. ROMEN RAKAMLARIYLA İLGİLİ İŞLEVLER

Romen rakamlarıyla ilgili işlem yapacak aşağıdaki işlevleri tanımlayın.

```
int rom_to_int(const char *str);
```

İşlev, kendisine gönderilen dizgedeki romen rakamlarıyla ifade edilmiş *int* türden sayı sınırları içinde kalan tamsayıyı, *int* türden bir tamsayıya çevirerek geri dönüş değeri olarak üretmeli.
Eğer adresi alınan yazı romen rakamlarıyla ifade edilebilen bir tamsayıyı göstermiyorsa, işlev *-1* değerine geri dönmeli.
Yazı boşluk karakterleriyle başlayabilir. Bu durumda boşluk karakterleri dikkate alınmayacaktır. Romen rakamları ile yazılan sayının, sonlandırıcı karakter görüldüğünde

ya da romen rakamları karakterleri dışında ilk karakterin görülmesi durumunda sonlandığı düşünülmeli.

```
char *int_to_rom (int number, char *str);
```

İşleve kendisine birinci argüman olarak gönderilen *int* türden bir tamsayıyı, romen rakamlarıyla ifade ederek, ikinci argüman olarak kendisine gönderilen adresten başlayarak yazmalı.

Birinci argüman romen rakamlarıyla ifade edilemeyecek değer ise işlev *NULL* adresine geri dönmeli. Aksi halde işlev kendisine gönderilen adrese geri dönmeli.

```
void display_rom (unsigned int number);
```

İşlev kendisine argüman olarak gönderilen sayıyı romen rakamlarıyla ekrana yazmalı.

```
int isvalid_rom(const char *str);
```

***str* içindeki yazının romen rakamlarıyla yazılmış geçerli bir sayı olup olmadığını sınar. Geçerli bir tamsayı ise işlev sıfır dışı bir değere, geçerli değil ise *0* değerine geri döner.**

```
I   1
V   5
X   10
L   50
C   100
D   500
M   1000
```

****

93. strlen İŞLEVİ

***strlen*, *string.h* başlık dosyası içinde bildirilen standart bir C işlevidir.**

```
size_t strlen(const char *string);
```

***strlen* işlevi başlangıç adresini aldığı yazının uzunluğu ile geri döner. İşlevi *mystrlen* ismiyle tanımlayarak aşağıdaki *main* işlevi ile sınayın:**

```
#include <stdio.h>

#define    SIZE    200

int mystrlen(const char *);

int main()
{
    char str[SIZE];

    printf("bir yazı giriniz : ");
    gets(str);
    printf("yazının uzunluğu = %d\n", mystrlen(str));

    return 0;
}
```

****

94. strchr İŞLEVİ

***strchr*, bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strchr(const char *string, int c);
```

İşlevin birinci parametresi gönderilen yazının başlangıç adresi, ikinci parametresi bir karakterin kod numarasıdır. İşlev yazı içinde *c* karakterini arar. İşlev ilk bulduğu *c* karakterinin adresiyle geri döner. Eğer yazı içinde *c* karakteri yoksa işlev *NULL* adresi ile geri döner. İşlevi *mystrchr* ismiyle tanımlayarak aşağıdaki *main* işlevi ile sınayın:

```
#include <stdio.h>

char *mystrchr(const char *, int );

int main()
{
    char string[] = "The quick brown dog jumps over the lazy fox";
    int ch;

    for (ch = 'k'; ch <= 'n'; ++ch)
        printf("%s\n", mystrchr(string, ch));

    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
k brown dog jumps over the lazy fox
lazy fox
mps over the lazy fox
n dog jumps over the lazy fox
```

****

95. strrchr İŞLEVİ

***strrchr*, bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strrchr(const char *string, int c );
```

İşlevin birinci parametresi gönderilen yazının başlangıç adresi, ikinci parametresi bir karakterin kod numarasıdır. İşlev yazının sonundan başlayarak, yazı içinde *c* karakterini arar. İşlev ilk bulduğu *c* karakterinin adresiyle geri döner. Eğer yazı içinde *c* karakteri yoksa, işlev *NULL* adresi ile geri döner. İşlevi *mystrchr* ismiyle tanımlayarak aşağıdaki kod ile sınayın:

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

char *mystrrchr(const char *ptr, int ch);

int main()
{
    char string[] = "12345678901234567890123456789012345678901234567890";
    int ch;

    for (ch = '0'; ch <= '9'; ++ch)
        printf("%s\n", mystrrchr(string, ch));

    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
0
1234567890
234567890
34567890
4567890
567890
67890
7890
890
90
```

****

96. strcpy İŞLEVİ

***strcpy*, bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strcpy(char *dest, const char *source);
```

İşlev, başlangıç adresi *source* olan yazıyı *dest* adresine kopyalar. İşlevin geri dönüş değeri *dest* adresidir. İşlevi *mystrcpy* ismiyle tanımlayarak aşağıdaki kod ile sınayın:

```
#include <stdio.h>

#define    SIZE       200

char *mystrcpy(char *dest, const char *source);

int main()
{
    char s1[SIZE];
    char s2[SIZE];

    printf("bir yazı girniz : ");
    gets(s1);

    mystrcpy(s2, s1);
    printf("kopyalanan yazı = %s!", s2);

    return 0;
}
```

****

97. strcat İŞLEVİ

***strcat* bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strcat(char *dest, const char *source);
```

İşlev, başlangıç adresi *source* olan yazının sonuna *dest* adresindeki yazıyı ekler. İşlevin geri dönüş değeri *dest* adresidir. İşlevi *mystrcat* ismiyle tanımlayarak, kendi yazacağınız bir *main* işleviyle sınayın.

****

98. strcmp İŞLEVİ

***strcmp* bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir:**

```
int strcmp(const char *s1, const char *s2);
```

İşlev *s1* ve *s2* adreslerindeki yazıları karşılaştırır. Eğer *s1* adresindeki yazı *s2* adresindeki yazıdan daha büyükse, işlev pozitif bir değere, *s1* adresindeki yazı *s2* adresindeki yazıdan daha küçükse 0'dan küçük bir değere, yazılar eşit ise 0 değerine geri döner. Bu işlevi *mystrcmp* ismiyle tanımlayın. Yazdığınız işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

#define    SIZE   200

int mystrcmp(const char *, const char*);

int main()
{
    char name1[SIZE];
    char name2[SIZE];
    int comp_result;

    printf("birinci ismi giriniz : ");
    gets(name1);
    printf("ikinci ismi giriniz  : ");
    gets(name2);
    comp_result = mystrcmp(name1, name2);

    if (comp_result > 0)
        printf("(%s) > (%s)\n", name1, name2);
    else if (comp_result < 0)
        printf("(%s) < (%s)\n", name1, name2);
    else
        printf("(%s) == (%s)\n", name1, name2);

    return 0;
}
```

****

99. stricmp İŞLEVİ

***stricmp* işlevi, standart olmamasına karşın pekçok C derleyicisinde bulunur. Bu işlev iki yazıyı büyük harf küçük harf ayrımı yapmadan karşılaştırır. Bu işlevi *mystricmp* ismiyle tanımlayın.**

```
int mystricmp(const char *s1, const char *s2);
```

İşlev birinci yazı ikinci yazıdan büyükse pozitif herhangi bir değere, küçükse negatif herhangi bir değere ve iki yazı birbirine eşitse sıfır değerine geri dönmeli.
Yazdığınız işlevi *strcmp* işlevini sınamakta kullandığınız *main* işlevi ile sınayabilirsiniz.

****

100. strcmp İŞLEVİ (Türkçe)

İki yazıyı Türkçe karakterleri dikkate alarak büyük harf küçük harf duyarlılığı ile ile karşılaştıran strcmptrk isimli işlevi tasarlayın:

```
int strcmptrk(const char *s1, const char *s2);
```

İşlev birinci yazı ikinci yazıdan büyükse pozitif herhangi bir değere, küçükse negatif herhangi bir değere, eşitse sıfıra geri dönmeli.
Yazdığınız işlevi *strcmp* işlevini sınamakta kullandığınız *main* işlevi ile sınayabilirsiniz.

****

101. strstr İŞLEVİ

***strstr*, bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strstr(const char *s1, const char *s2);
```

İşlev başlangıç adresi *s1* olan yazı içinde başlangıç adresi *s2* olan yazıyı arar. Eğer aranılan yazı bulunursa, işlevin geri dönüş değeri bulunan yerin adresidir. Eğer yazı bulunmazsa işlev *NULL* adresine geri döner. İşlevi *mystrstr* ismiyle tanımlayın. Yazdığınız işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

char *mystrstr(const char *s1, const char *s2);

int main()
{
    char s1[] = " İyi bir C programcısı olmak için çok çalışmak
gerekir!..";
    char s2[] = "çok";

    char *p = mystrstr(s1, s2);
    printf("%s\n", p);      /* çok çalışmak gerekir!.. */

    return 0;
}
```

****

102. strncpy İŞLEVİ

***strncpy* bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strncpy(char *dest, const char *source, size_t n);
```

İşlev *source* adresindeki yazının ilk *n* karakterini *dest* adresine kopyalar. Eğer *n* değeri *source* adresindeki yazının uzunluğuna eşit ya da yazının uzunluğundan küçük ise, kopyalama yapılan yazının sonuna sonlandırıcı karakter eklenmez. *n* değeri *source* adresindeki yazının uzunluğundan daha büyükse, kopyalama yapılan yazının sonuna *source* adresindeki yazının kopyalanmasından sonra, sonlandırıcı karakterler eklenmeli.
İşlev *dest* adresine geri dönmeli.
Bu işlevi *mystrncpy* ismiyle tanımlayın.
Yazdığınız işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

int main()
{
    char str[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char dest[27];
    int k;

    for (k = 1; k <= 5; ++k) {
```

```
        mystrncpy(dest, str, k);
        dest[k] = '\0';
        printf("(%s)\n", dest);
    }
    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
(A)
(AB)
(ABC)
(ABCD)
(ABCDE)
```

****

103. strncat İŞLEVİ

***strncat*, bildirimi *string.h* başlık dosyası içinde bulunan standart bir C işlevidir.**

```
char *strncat(char *dest, const char *source, size_t n);
```

İşlev *dest* adresindeki yazının sonuna *source* adresindeki yazının ilk *n* karakterini ekler. *source* adresindeki yazının ilk karakteri dest adresindeki yazının sonundaki sonlandırıcı karakterin üzerine yazılır. Eğer *n* değerine ulaşılmadan source adresindeki yazıda sonlandırıcı karakter ile karşılaşılırsa, dest adresindeki yazının sonuna, yalnızca sonlandırıcı karaktere kadar olan karakterler eklenir. *dest* adresindeki yazının sonuna her zaman sonlandırıcı karakter eklenir. İşlevin geri dönüş değeri *dest* adresidir.
Bu işlevi *mystrncat* ismiyle tanımlayın.
Yazdığınız işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

char *mystrncat(char *, const char *, size_t);

#define    SIZE 200

int main()
{
    char s1[SIZE];
    char s2[SIZE];
    int n;

    printf("birinci yaziyi girin : ");
    gets(s1);
    printf("ikinci yaziyi girin : ");
    gets(s2);
    printf("birinci yazinin sonuna kac karakter eklemek istiyorsunuz? : ");
    scanf("%d", &n);
    printf("eklemeden once = (%s)\n", s1);
    mystrncat(s1, s2, n);
    printf("eklemeden sonra = (%s)\n", s1);

    return 0;
}
```

****

104. strrev İŞLEVİ

strrev **işlevi standart bir C işlevi olmamasına karşın pekçok C derleyicisinde bulunur:**

```
char *mystrrev(char *str);
```

Bu işlev başlangıç adresini aldığı yazıyı ters çevirir. Bu işlevi *mystrrev* ismiyle tanımlayın. İşlevin geri dönüş değeri aldığı yazının başlangıç adresidir.

```
#include <stdio.h>

#define    SIZE   100

char *mystrrev(char *str);

int main()
{
    char s[SIZE] = "Life is very short and there's no time for fighting";

    mystrrev(s);
    printf("%s!\n", s);

    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
gnithgif rof emit on s'ereht dna trohs yrev si efiL!
```

****

105. strset İŞLEVİ

strset **işlevi standart bir C işlevi olmamasına karşın hemen hemen tüm C derleyicilerinde bulunur. Bu işlev başlangıç adresini aldığı yazının tüm karakterlerini ikinci parametresine sıra numarasını aldığı karaktere dönüştürür. İşlevin geri dönüş değeri aldığı yazının başlangıç adresidir. Bu işlevi *mystrset* ismiyle tanımlayın:**

```
char *mystrset(char *str);
```

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

int main()
{
    char str[] = "Necati";

    mystrset(str, 'X');
    printf("(%s)\n", str); /* (XXXXXX) yazmali */

    return 0;
}
```

****

106. squeze İŞLEVİ

Aşağıda bildirimi verilen işlevi tanımlayın:

```
char *squeze(char *s1, const char *s2);
```

***squeze* işlevi *s1* adresindeki yazı içindeki karakterlerden biri, s2 adresi içinde bulunan yazı içinde var ise bu karakteri yazıdan silmeli. Başka bir deyişle *s1* adresindeki yazı içinde *s2* adresindeki yazıdaki karakterlerden hiçbirini içermeyecek hale getirilmeli. İşlevin *s1* adresine geri dönmeli.**

```
#include <stdio.h>

#define SIZE       100

char *squeze(char *s1, const char *s2);

int main()
{
    char s1[SIZE] = "Necati Ergin";
    char s2[SIZE] = "Nuri Yilmaz";

    printf("%s\n", squeze(s1, s2));

    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
ectEgn
```

****

107. TAMSAYIYI YAZIYA DÖNÜŞTÜRMEK

Verilen *int* türden bir sayının Türkçe metin karşılığını bir adresten başlayarak yerleştiren *numtotext* isimli işlevi tanımlayın. İşlevin bildirimi aşağıdaki gibidir:

```
char *numtotext(long number, char *str);
```

Birinci parametresi yazıya çevrilecek olan sayıyı, ikinci parametresi ise yazının yerleştirileceği dizinin başlangıç adresini göstermektedir. İşlev *str* adresine geri dönmeli. Metin küçük harflerden oluşmalı. Yazdığınız işlevi aşağıdaki *main* işlevi ile sınayabilirsiniz:

```
#include <stdio.h>

#define SIZE       100

char *numtotext(long number, char *str);

int main()
{
    char str[SIZE];

    numtotext(1345417L, str);
    printf("%s\n", str);   /* birmilyonüçyüzkırkbeşbindörtyüzonyedi */

    return 0;
}
```

****

108. YAZIDAKİ KARAKTERİ BASKA BİR KARAKTERLE DEĞİŞTİRMEK

Bir yazı içindeki istenen bir karakteri bir başka karakterle değiştiren aşağıdaki işlevi yazın:

```
char *replace_chars(char *str, char x, char y);
```

İşlevin birinci parametresi karakter dizisinin başlangıç adresi, ikinci parametresi değiştirilecek olan karakter, üçüncü parametresi de hangi karakterle değiştirileceğidir. Geri dönüş değeri birinci parametresiyle belirtilen adresin kendisidir.

```
#include <stdio.h>

char *replace_chars(char *str, char x, char y);


int main()
{
    char s[ ] = "Bu bir denemedir!";

    puts(replace(s, 'e', 'x'));

    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
/* Bu bir dxnxmxdir! */
```

****

109. YAZIDAKİ KÜÇÜK HARFLERİ BÜYÜK HARFE BÜYÜK HARFLERİ KÜÇÜK HARFE DÖNÜŞTÜRMEK

Bir yazının içindeki küçük harfleri büyük harfe, büyük harfleri ise küçük harfe çeviren

```
char *change_case(char *str);
```

işlevini tanımlayın. İşlev *str* adresindeki yazıdaki küçük harfleri büyük harfe, büyük harfleri küçük harfe çevirmeli. İngilizce harf karakterleri dışındaki harflerde bir değişiklik yapılmamalı. İşlevin kendisine gönderilen yazının başlangıç adresini geri döndürmeli.

****

110. YAZIDAN BAŞKA BİR YAZIYI SİLMEK

Bir yazının içinden başka bir yazıyı (her yinelenmesinde) silen, *remove_str* isimli işlevi tanımlayın:

```
char *remove_str(char *p1, const char *p2);
```

İşlev *p1* adresindeki yazının içinde *p2* adresindeki yazıyı siler. Eğer *p1* adresindeki yazının içinde *p2* adresindeki yazı birden fazla kez yineleniyorsa, yinelenen tüm yazılar silinmeli. İşlev, içinde silme işlemi yapılan yazının adresine geri dönmeli. Aşağıdaki sınama kodunu kullanabilirsiniz:

```
#include <stdio.h>

char *remove_str(char *p1, const char *p2);

int main()
```

```
{
    char s1[100];
    char s2[100];

    printf("bir yazi girin: ");
    gets(s1);
    printf("silinecek yaziyi girin: ");
    gets(s2);
    printf("silme isleminden once : (%s)\n", s1);
    remove_str(s1, s2);
    printf("silme isleminden sonra : (%s)\n", s1);

    return 0;
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
bir yazi girin: deniz sizi izliyor
silinecek yaziyi girin: iz
silme isleminden once : (deniz sizi izliyor)
silme isleminden sonra : (den si liyor)
```

****

111. YAZIDAN BOŞLUK KARAKTERLERİNİ SİLEN İŞLEV

Bir yazının içindeki boşluk karakterlerini silecek *rem_space* isimli işlevi tanımlayın:

```
char *rem_space(char *str);
```

İşlev, kendisine gönderilen adresteki yazıdaki tüm boşlukları (*white space* karakterleri) silerek yazıyı aynı adrese boşluksuz olarak yerleştirmeli. Yazının sonunda yine sonlandırıcı karakter bulunmalı.
İşlevin geri dönüş değeri işleve gönderilen adres olmalı.

```
#include <stdio.h>

#define SIZE       100

int main()
{
    char s[SIZE];

    printf("bir yazi giriniz : ");
    gets(s);
    rem_space(s);
    printf("yazinin yeni sekli = %s", s);

    return 0;
}
```

Yukarıdaki kod parçasında gets işlevi çağırıldığında klavyeden aşağıdaki yazının girildiğini düşünelim (_ boşluk karakterlerini gösteriyor!)

```
__İy_i__bi__r__C_P_r_o_gr_a_m_c_ısı__olmak__ci__n_ok_çalışmak_gere_kir____!
```

***puts* işlevi çağrıldığında ekrana yazılan yazı şu şekilde olmalıdır:**

```
İyibirCProgramcısıolmakiçinçokçalışmakgerekir!
```

****

112. YAZIDAN YİNELENEN KARAKTERLERİN SİLİNMESİ

Bir yazının içinden yinelenen karakterleri silen *remove_duplicate* isimli işlevi tanımlayın:

```
char *remove_duplicate(char *);
```

İşlev bir yazının başlangıç adresini alarak, yazıdan yinelenen karakterleri silmeli. Yani işlev çağrıldıktan sonra yazıda aynı karakterden birden fazla olmamalı.
İşlevin yazının içinden silme işlemi yaptığı yazının başlangıç adresine dönmeli.

```
#include <stdio.h>

int main()
{
    char str[SIZE];

    printf("bir yazi giriniz : ");
    gets(str);

    remove_duplicate(str);
    printf("(%s)\n", str);
    return 0;
}
```

Örnek:

```
bir yazi giriniz : kahramanmaras
(kahrmns)
```

****

113. YAZI BAŞINDAKİ VE SONUNDAKİ BOŞLUKLARI SİLEN İŞLEV

Bir yazının başındaki ve sonundaki boşluk karakterlerini atarak başka bir diziye yerleştiren *trim* isimli işlevi yazın. İşlevin bildirimi aşağıdaki gibidir:

```
char *trim(char *dest, const char *source);
```

İşlevin ilk parametresi yerleştirilecek yazının adresi, ikinci parametresi ise boşluklu yazının adresidir. İşlevin geri dönüş değeri yerleştirme işleminin yapıldığı adrestir.

```
#include <stdio.h>

#define SIZE   128

char *trim(char *dest, const char *source);

int main()
{
    char s[] = "   Iyi bir C programcisi olmak için çok çalismak gerekir
";
    char d[128];

    trim(d, s);
    printf("Elde edilen yazi: (%s)\n", d);

    return 0;
```

```
}
```

Programın ekran çıktısı aşağıdaki gibi olmalı:

```
Elde edilen yazi: (Iyi bir C programcisi olmak için çok çalışmak gerekir)
```

****

114. YAZININ SONUNDA DİĞER YAZI VAR MI?

Bir yazının sonunda başka bir yazının olup olmadığını sınayan *strend* isimli işlevi tanımlayın:

```
int strend(const char *s1, const char *s2);
```

İşlev *s1* yazısının sonunda *s2* yazısı varsa sıfır dışı bir değere, aksi halde *0* değerine geri dönmeli.

```
#include <stdio.h>

#define    SIZE   100

int strend(const char *s1, const char *s2);

int main()
{
    char s1[SIZE];
    char s2[SIZE];

    printf("kaynak yaziyi giriniz : ");
    gets(s1);
    printf("kaynak yazinin sonunda aranacak yaziyi giriniz : ");
    gets(s2);
    if (strend(s1, s2))
        printf("(%s) yazisinin sonunda (%s) yazisi var!\n", s1, s2);
    else
        printf("(%s) yazisinin sonunda (%s) yazisi yok!\n", s1, s2);

    return 0;
}
```

****

115. GÖSTERİCİ DİZİLERİ

Aşağıda *pnames* isimli *200* elemanlı bir gösterici dizisi veriliyor:

```
const char *const pnames[200] = {"Muhittin", "Mufit", "Serkan", "Kaan",
"Izzet", "Muzaffer", "Umid", "Sami", "Uykura", "Kayhan", "Yakup", "Mert",
"Cetin", "Ilknur", "Gokhan", "Salah", "Irem", "Korhan", "Isin", "Berk",
"Tacettin", "Duygu", "Figen", "Funda", "Fuat", "Arda","Muhsin", "Guray",
"Necati", "Kadriye", "Gurbuz", "Hamide", "Volkan", "Selami", "Esra",
"Cumhur", "Can", "Aleyna", "Salih", "Kamuran", "Ferhan", "Furuze", "Suku
fe", "Sidre", "Cahit", "Sercan", "Jale", "Bilge", "Tijen", "Tufan",
"Zeliha", "Cevdet", "Burak", "Ufuk", "Zarife", "Bengisu", "Melda",
"Zafer", "Yelda", "Didar", "Hande", "Cumali", "Dilek", "Kemal", "Hasan",
"Ismail", "Fatma", "Burcu", "Siyami", "Busra", "Gizem", "Ramize", "Sezai",
"Polat", "Melike", "Umut", "Serencan", "Uzeyir", "Kamile", "Deniz",
"Veysel", "Emrehan", "Gulcan", "Ender", "Furkan", "Veli", "Muhtesem",
"Necmettin", "Osman", "Perihan", "Didem", "Tunc", "Irmak","Petek",
"Sebahat", "Halime", "Mahir", "Leyla", "Eda", "Ali", "Ibrahim", "Berivan",
```

```
"Ilter", "Metin", "Bozkurt", "Olgun", "Hudai", "Esin", "Taylan", "Memduh",
"Selimcan", "Ayse", "Baran", "Keriman", "Tankurt", "Feramuz", "Abdullah",
"Ferda", "Kuntay", "Ceyhan", "Edip", "Nuri", "Mahide", "Selda",
"Murathan", "Celebi", "Talha", "Mehtap", "Ece", "Abdurrahman", "Baki",
"Anil", "Unsal", "Melahat", "Sezgin", "Miran", "Cansel", "Sibel", "Aliye",
"Derya", "Naime", "Gulden", "Konar","Soner", "Gulistan", "Mestan",
"Galip", "Lamia", "Hakan", "Ugur", "Ege", "Husnu", "Sumbul", "Mustafa",
"Simge", "Selva", "Mehmet", "Irfan", "Cezmi", "Turgut", "Konur",
"Aysegul", "Mesut", "Ceren", "Adem", "Ferhat", "Murat", "Yunus", "Beyhan",
"Erdem", "Sunusi", "Ata", "Sencer", "Handan", "Cihan", "Levent", "Aykut",
"Ferit", "Demir", "Kerim", "Feraye", "Caner", "Baris", "Ozlem", "Ihsan",
"Meltem","Kurthan", "Okan", "Huseyin", "Sezgi", "Celik", "Cemal", "Sinan",
"Sefa", "Cigdem", "Leman", "Nejla", "Gurkan", "Gulsum", "Damla",};
```

Bu gösterici dizisini kullanarak sırasıyla aşağıdaki işlemleri yapın:

1. Dizinin elemanlarının göstermekte olduğu isimleri ekrana yazdırın.
2. Dizinin elemanlarının göstermekte olduğu isimlerin ilk harfini ekrana yazdırın.
3. Dizinin elemanlarının göstermekte olduğu isimlerin son harfini ekrana yazdırın.
4. Kullanıcı klavyeden bir harf karakteri girsin. Dizinin elemanlarının göstermekte olduğu İsimlerden, içinde bu harf olanları ekrana yazdırın. (küçük ya da büyük harf olabilir)
5. Kullanıcı klavyeden bir yazı karakteri girsin. Dizinin elemanlarının göstermekte olduğu isimlerden, içinde bu yazı olanları ekrana yazdırın. (küçük ya da büyük harf olabilir)
6. Kullanıcı klavyeden bir isim girsin. Klavyeden girilen ismin, dizi elemanlarından herhangi birinin gösterdiği yazılardan biri olup olmadığını sınayın. Eğer elemanlardan birinin gösterdiği yazılardan biriyse, bu yazıyı gösteren dizi elemanının indisini ekrana yazdırın.
7. Diziyi, elemanlarının gösterdiği yazılar küçükten büyüğe olacak biçimde sıralayın.
8. Dizinin gösterdiği yazıların uzunluklarının aritmetik ortalamasını ekrana yazdırın.
9. Diziyi elemanların gösterdiği yazılarının uzunlukları küçükten büyüğe olacak biçimde sıralayın. Uzunluğu eşit olan yazılar da kendi aralarında alfabetik olarak sıralanmalı.

****

116. strdup İŞLEVİ

***strdup* işlevi standart olmamasına karşın derleyici paketlerinin çoğuyla sunulur:**

```
char *strdup(const char *ptr);
```

İşlev *ptr* adresindeki yazının bir kopyasını dinamik bir bellek alanına kopyalar. İşlevin geri dönüş değeri kopya yazının başlangıç adresidir. Bu işlevi *mstrdup* ismiyle tanımlayın. Tanımladığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

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

char *mstrdup(const char *ptr);

int main()
{
    char str[100];
    char *pd;

    printf("bir yazı girin : ");
    gets(str);

    pd = mstrdup(str);

    printf("(%s)\n", str);
    printf("(%s)\n", pd);
```

```
    free(pd);

    return 0;
}
```

****

117. İKİ YAZIYI BİRLEŞTİRMEK

Okuma amacıyla adresini aldığı iki yazıyı dinamik bir bellek alanında birleştiren strcon isimli işlevi tanımlayın:

```
char *strcon(const char *p1, const char *p2);
```

İşlev *p1* ve *p2* adreslerindeki yazıları dinamik bir bellek alanında birleştirerek, birleştirilmiş yazının başlangıç adresine geri dönmeli.

Tanımladığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

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

char *strcon(const char *p1, const char *p2);

int main()
{
    char s1[100];
    char s2[100];
    char *pd;

    printf("birinci yazıyı girin : ");
    gets(s1);
    printf("ikinci yazıyı girin : ");
    gets(s2);

    pd = strcon(s1, s2);

    printf("(%s)\n", s1);
    printf("(%s)\n", s2);
    printf("(%s)\n", pd);

    free(pd);

    return 0;
}
```

****

118. merge İŞLEVİ

***int* türden sırasız iki diziyi, küçükten büyüğe doğru sıralı bir biçimde yerleştiren *merge* isimli işlevi tanımlayın:**

```
int *merge(const int *p1, int size1, const int *p2, int size2);
```

İşleve gönderilecek argümanlar sırasız olan dizilerin başlangıç adresleri ve uzunluklarıdır. İşlev kendi içinde iki dizinin toplam uzunluğu kadar bir bellek bloğunu dinamik olarak elde ederek birleştirme işlemini dinamik olarak yaratılan blokta gerçekleştirmeli. İşlev, dinamik dizinin başlangıç adresiyle geri dönmeli.

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

int *merge(const int *p1, int size1, const int *p2, int size2);

int main()
{
    int a[10] = {1, 4, 7, 2, 3, 2, 1, 8, 7, 6};
    int b[8] = {0, 4, 0, 3, 1, 9, 6, 4};
    int *ptr, k;

    ptr = merge(a, 10, b, 8);
    printf("birlestirilmis dizi :\n");

    for (k = 0; k < 18; ++k)
        printf("%d ", ptr[k]);

    free(ptr);

    return 0;
}
```

****

119. unique_copy İŞLEVİ

Bildirimi aşağıda verilen *unique_copy* isimli işlevi tanımlayın:

```
int *unique_copy(const int *ptr, int size);
```

***int* türden bir dizinin adresini ve boyutunu alan işlev, dizideki değerleri dinamik olarak elde ettiği bir alana kopyalamalı. Ancak dizide bulunan bir değere dizi içinde yeniden rastlanırsa bu değer dinamik alana kopyalanmamalı. Yani dinamik alandaki her değerden bir tane bulunmalı.**

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

#define SIZE 100

int *unique_copy(const int *ptr, int size);

int main()
{
    int a[SIZE];
    int k, *ptr;
    srand(time(0));

    for (k = 0; k < SIZE; ++k)
        a[k] = rand() % 20;
    ptr = unique_copy(a, SIZE);
    printf("dizi yazdiriliyor \n");

    for (k = 0; k < SIZE; ++k)
        printf("%3d ", a[k]);

    printf("\n\ndinamik dizi yazdiriliyor \n");
    for (k = 0; k < SIZE; ++k)
        printf("%3d ", ptr[k]);
    free(ptr);

    return 0;
}
```

****

120. KOMUT SATIRI ARGÜMANLARINI TERSTEN YAZDIRAN PROGRAM

Komut satırından aldığı argümanları ekrana ters sırada yazdıran bir C programı yazın. Programın isminin "*tersyaz*" olduğunu düşünelim. Program komut satırından

```
tersyaz ahmet hasan mehmet ali
```

biçiminde çalıştırıldığında ekran çıktısı

```
ali mehmet hasan ahmet
```

olmalı

****

121. KOMUT SATIRINDAN GİRİLEN TAMSAYILARIN ORTALAMASI

Komut satırından girilen tamsayıların aritmetik ortalamasını ekrana yazdıran "ao.c" isimli programı yazın. Program komut satırından aşağıdaki biçimde çalıştırılmalı:

```
ao 25 37 134 98 87 -12 126
ortalama =  70,714286
```

****

122. NOKTA DAİRENİN İÇİNDE Mİ? NOKTA KARENİN İÇİNDE Mİ?

***Point* isimli bir yapımız var:**

```
typedef struct {
    int m_x, m_y
}Point;
```

Bu yapı grafik ekrandaki bir pixelin *x* ve *y* koordinat değerlerini tutmaktadır. *Point* yapı nesnelerinden oluşan bir dizi üzerinde aşağıdaki işlemleri yapan işlevleri yazın:

```
Bool is_circle(const Point *ptr, size_t size);
```

İşlev *Point* dizisi içindeki noktaların bir çember olusturup oluşturmadıklarını sınamalı.

```
Bool is_square(const Point *ptr, size_t size);
```

Bu işlev *Point* dizisi içindeki noktaların bir kare oluşturup oluşturmadıklarını sınamalı.

****

123. YAPI DİZİLERİ ÜZERİNDE İŞLEM YAPAN İŞLEVLER

Aşağıda bildirimi verilen *Person* yapısını kullanarak *n* kişinin bilgilerini klavyeden alan bunları isim ve soyadlarına göre sıraya dizen ve ekrana yazdıran işlevleri tasarlayın:

```
#define MAX_NAME_LEN       20
#define MAX_FNAME_LEN      30

typedef struct _Date {
    int day, month, year;
}Date;

typedef struct _Person {
    char name[MAX_NAME_LEN + 1];
    char fname[MAX_FNAME_LEN + 1];
    Date bdate;
}Person;
```

Klavyeden bilgileri alarak diziye yerleştiren işlevin bildirimi şöyledir:

```
void get_info(Person *parray, size_t size);
```

Bu işlev parametre değişkeni olarak yapı dizisinin başlangıç adresini ve dizinin boyutunu almaktadır. Diziyi isim ve soyadlarına göre sıraya dizen işlevin bildirimi şöyledir:

```
void sort_info(Person*ptr, size_t size);
```

İşlevin parametre değişkenleri dizinin başlangıç adresi ve uzunluğudur.
Şahıs bilgilerini ekrana yazan işlevin bildirimi ise:

```
void display_info(const Person *ptr, size_t size);
```

****

124. BİTSEL İŞLEÇL ALIŞTIRMALARI (1)

Aşağıdaki programda *printf* çağrılarının ekrana yazdıracağı değerleri programı çalıştırmadan bulmaya çalışın:

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

```
int main()
{
    int x, y, z;

    x = 3;
    y = 2;
    z = 1;

    printf("%d\n", x | y & z);
    printf("%d\n", x | y & ~z);
    printf("%d\n", x ^ y & ~z);
    printf("%d\n", x & y && z);

    x = 1;
    y = -1;

    printf("%d\n", !x | x);
    printf("%d\n", ~x | x);
    printf("%d\n", x ^ x);

    x <<= 3;
    printf("%d\n", x);

    y <<= 3;
    printf("%d\n", y);

    return 0;
}
```

****

125. 2'NİN KUVVETİ Mİ

***unsigned int* türden bir değerin *2*'nin kuvveti olup olmadığını sınayan bir kod parçası yazın. Bir tamsayı 2'nin belirli bir tamsayı üssü ise yalnızca bir biti *1* demektir. Bir döngü içinde söz konusu nesnenin sırayla bütün bitlerine erişip, söz konusu sınamayı yapabilirsiniz. Ancak bir döngü deyimi kullanmadan da, daha verimli bir kod parçası yazmak mümkündür.**
Yazdığınız kod parçasının taşınabilir özellikte olmalı. *int* türü uzunluğunun sistemden sisteme değişebileceğini unutmayın.

****

126. BİTSEL DÖNDÜRME İŞLEMİ YAPAN İŞLEVLER

Aşağıda bildirimleri verilen işlevleri tanımlayarak kendi yazacağınız bir kod ile sınayın:

```
int rotate_left(int number, int n);
```

***rotate_left* işlevi kendisine birinci argüman olarak gönderilen *int* türden değeri, ikinci argüman olan pozisyon kadar sola kaydırararak elde edilen tamsayıya geri dönmeli. Her bir sola kaydırma işleminde, sayının sağından yapılacak besleme menzil dışına çıkacak bit değeri ile aynı olmalı. Yani menzil dışına çıkacak bit *1* ise sağdan *1* biti ile, menzil dışına çıkacak bit *0* ise sağdan *0* biti ile besleme yapılmalı.**

```
int rotate_right(int number, int n);
```

***rotate_right* işlevi kendisine birinci argüman olarak gönderilen *int* türden değeri, ikinci argüman olan pozisyon kadar sağa kaydırararak elde edilen tamsayıya geri dönmeli. Her bir sağa kaydırma işleminde, sayının solundan yapılacak besleme menzil dışına çıkacak**

bit değeri ile aynı olmalı. Yani menzil dışına çıkacak bit *1* ise sağdan *1* biti ile, menzil dışına çıkacak bit *0* ise sağdan *0* biti ile besleme yapılmalı.

****

127. BİTSEL İŞLEÇLERLE MUTLAK DEĞER ALMA

***x, int* türden bir değişken olsun. Bitsel işleçleri kullanarak öyle bir ifade yazın ki, ifadenin değeri x in mutlak değeri olsun.**

****

128. BİTSEL İŞLEÇLERLE SİGNUM DEĞERİ ELDE ETME

***x int* türden bir değişken olsun. Bitsel işleçleri kullanarak öyle bir ifade yazın ki, ifadenin değeri**

```
x > 0 ise 1
x == 0 ise 0
x < 0 ise -1 olsun.
```

****

129. BİTSEL İŞLEÇLERLE YAPILACAK KARŞILAŞTIRMA

Kendisine gönderilen *int* türden iki tamsayıyı, koşul işlecini ve C dilinin kontrol deyimlerini kullanmadan, yalnızca bitsel işleçleri kullanarak kıyaslayan *compare* isimli işlevi yazın:

```
int compare(int number1, int number2);
```

***compare* işlevi ilk parametre ile geçilen sayı büyükse işlev pozitif bir değere, küçükse negatif bir değere, sayılar birbirlerine eşit ise *0* değerine geri dönmeli.**

****

130. TAMSAYIYI İKİLİK SAYI SİSTEMİNDE YAZDIRACAK İŞLEV

Kendisine gönderilen *int* türden bir tamsayı değeri ekrana ikilik sayı sisteminde yazdıracak *print_binary* isimli işlevi tanımlayın:

```
void print_binary(int val);
```

Yazdığınız işlevi aşağıdaki *main* işleviyle sınayabilirsiniz:

```
#include <stdio.h>

void print_binary(int val);

int main()
{
    int k;

    for (k = 17800; k <= 17820; ++k) {
        printf(“%d = “);
        print_binary(k);
        putchar(‘\n’);
    }
    return 0;
}
```

****

131. DOSYA KOPYALAYAN PROGRAM

Çalıştırıldığında bir dosyanın yeni bir kopyasını oluşturan *filecopy.c* isimli programı yazın. Program komut satırından aşağıdaki gibi çalıştırılmalı:

```
filecopy kaynak_dosya_ismi   yeni_dosya_ismi
```

****

132. DOSYA ŞİFRELEYEN PROGRAM

Komut satırından çalışacak aşağıdaki şifreleme ve açma programını yazın:

```
<enc> <filename> <key>
```

Birinci komut satırı argümanı şifrelenecek dosyanın ismi, ikinci komut satırı argümanı anahtar değeridir.
***"enc"* programı, komut satırından verilen anahtar değerinden üreteceği bir değeri standart *srand* işlevine geçerek tohum değeri olarak kullanmalı. Dosyanın her bir *byte*'ı standart *rand* işlevine yapılan çağrılardan elde edilen değerlerle bitsel özel veya işlemine sokulmalı.**
Aynı anahtar değeri kullanılırsa "*enc*" programı şifrelenmiş dosyayı aynı işlemlerle açmalı.

****

133. DOSYALARI BÖLMEK VE BİRLEŞTİRMEK

Bir dosyayı verilen büyüklükte parcalara ayıran "bol.c" isimli programı, ve parcalara ayrılmış dosyayı yeniden birleştirerek eski hale getiren "bir.c" isimli programı yazınız.

"bol.c" programı komut satırından aşağıdaki gibi çalıştırılmalı:

```
bol deneme.exe 2000
```

Birinci komut satırı argümanı, yani yukarıdaki örnekte "*dce.exe*" parcalara bölünecek dosyanın ismidir. İkinci komut satırı argümanı, yani yukarıdaki örnekteki *2000*, her bir parça dosyanın büyüklüğüdür. "bol" programının oluşturduğu parça dosyaların ismi aşağıdaki formata uymalı:

```
isim0001.ppp
```

Yukarıdaki isimlendirme formatında isim dosya isminin (uzantı olmadan) ilk 4 harfidir. Eğer dosya ismi 4 karakterden daha kısa ise dosya isminin tüm uzunlugu kullanılmalı. İzleyen 4 karakter parçanın numarasıdır. Parca dosyaların uzantısı "ppp" olmalı.

Örnek olarak "bol" programı ile *8765 byte* uzunluğunda "*deneme.exe*" isimli dosyanın *2000 byte*'lık parçalara bölündüğünü düşünelim. Programın üreteceği dosyalaraın isimleri ve büyüklükleri aşağıdaki gibi olmalı:

```
dene0001.ppp   2000 byte
dene0002.ppp   2000 byte
dene0003.ppp   2000 byte
dene0004.ppp   2000 byte
dene0005.ppp    765 byte
```

"*bir.c*" programı ise komut satırından aşağıdaki gibi çalıştırılmalı:

```
bir yeni.exe
```

Komut satırı argümanı, yani yukarıdaki örnekteki "*yeni.exe*" parçaların birleştirilmesiyle elde edilecek dosyanın ismidir. "bir" programı, *isim0001.ppp* formatına göre isimlendirilmiş parça dosyaları birleştirerek "yeni.exe" isimli dosyayı elde etmeli. Birleştirme işleminde sonra parça dosyalar silinmiş olmalı.

****

134. BİNARY DOSYANIN İÇERİĞİNİ YAZDIRMAK

Binary bir dosyanın içeriğini *byte byte* bir metin dosyasına onaltılık sayı sisteminde yazacak *typehex.c* isimli programı yazın. "*typehex.c*" isimli program komut satırından aşağıdaki gibi çalıştırılmalı:

```
typehex deneme.exe
```

Komut satırı argümanı, yani yukarıdaki örnekteki "*deneme.exe*" onaltılık sayı sisteminde yazdırılacak dosyanın ismidir. Program kaynak dosya ile aynı isme sahip ancak uzantısı "*hex*" olan bir dosya yaratmalı. Yukarıdaki örnekte yaratılacak dosyanın ismi "*deneme.hex*" olmalı. "*deneme.hex*" isimli metin dosyasının içeriği aşağıdaki düzende olmalı:

```
AB  12  05  23  36  FE  12  88  12  CD  FF  01  04  D0  EF  DD
23  12  25  36  74  A4  3C  8F  61  41  28  08  90  06  12  B8
AC  26  34  06  12  76  45  FE  3E  2A  7C  35  AC  D2  D0  DD
24  22  43  66  77  88  90  10  21  32  45  67  87  93  20  20
```

****

135. İŞLEV GÖSTERİCİLERİ (1)

Kabarcık sıralaması (*bubble sort*) algoritmasını kullanarak türden bağımsız sıralama yapan aşağıdaki işlevi yazın:

```
void bsort (void *base, unsigned  nelem,  nsigned width, int (*compare)
(const void *, const void *));
```

Kabarcık sıralaması yan yana iki elemanı karşılaştırarak yer değiştirme temeline dayanır. İşlev parametreleri şunlardır:

```
void *base
```

Sıralanacak dizinin başlangıç adresi

```
unsigned nelem
```

Dizinin eleman sayısı

```
unsigned width
```

Dizinin bir elemanının uzunluğu
Karşılaştırma işlevinin başlangıç adresi. bsort her karşılaştırma sırasında dizinin iki elemanını parametre olarak yerleştirerek bu işlevi çağırır. Küçükten büyüğe sıraya dizmek için karşılaştırma işlevi birinci eleman ikinciden büyükse pozitif bir değere, ikinci eleman birinciden büyükse negatif bir değere, iki eleman birbirine eşitse *0* değerine geri dönmeli.

```
#include <stdio.h>

#define SIZE       10
```

```
int cmp(const void *p1, const void *p2)
{
    return *(int *) p1 - *(int *) p2;
}

int main()
{
    int a[SIZE] = { 4, 7, 2, 8, 56, 4, 90, 23, 6, 10};
    int i;

    bsort(a, 10, sizeof(int), cmp);
    for (i = 0; i < SIZE; ++i)
        printf("%d\n", a[i]);

    return 0;
}
```

****

136. ÖZYİNELEMELİ İŞLEVLER (1)

Seçerek sıralama (*selection sort*) algoritmasını özyinelemeli bir işlev ile gerçekleştirin. İşlevin bildirimi aşağıdaki gibi olmalı:

```
void ssort(int *pArray, unsigned size);
```

İşlev *0* ile *size – 1* indisleri arasındaki elemanlardan en büyüğünü bularak *size – 1* indisindeki elemanla yer değiştirir. Sonra kendi kendini aşağıdaki gibi çağırır:

```
ssort(pArray, size – 1)
```

İşlem *size* değişkeninin değeri *1* oluncaya kadar sürer.

```
#include <stdio.h>

void ssort(int *pArray, unsigned size);

#define SIZE     10

int main()
{
    int a[10] = {3, 7, 3, 9, 5, 12, 31, 45, 1, 22};
    int i;

    ssort(a, 10);

    for (i = 0; i < SIZE; ++i)
        printf("%d\n", a[i]);

    return 0;
}
```

****

137. SEKİZ VEZİR PROBLEMİ

Bir satranç tahtasına birbirini almayacak şekilde *8* vezir yerleştirilebilir. Bir satranç tahtasına birbirini almayacak şekilde *8* veziri yerleştiren bir *C* programı yazın. Program vezirlerin konumlarını ekrana yazdırmalı. (*a5, b3 vs*.)

Program mümkün olan tüm çözümleri ekrana yazmalı.

****

138. POSIX SCANDİR İŞLEVİNİN TASARIMI

***POSIX* sistemlerinde kullanılan scandir isimli işlevi DOS ya da Windows ortamı için aşağıdaki biçimde tasarlayın:**

```
struct ffblk *scandir(const char  *path, int *pcount, int (*fcmp)(const
struct ffblk *));
```

- İşlev birinci parametresiyle belirtilen path içerisinde üçüncü parametresiyle belirtilen koşulu sağlayan dosyaları bulur; bu dosyaları dinamik olarak büyütülen bir diziye yerleştirir ve dinamik dizinin başlangıç adresiyle geri döner. İşlevin ikinci parametresi koşulu sağlayan dosyaların sayısının yerleştirileceği adresi belirtmektedir. Programcı ne kadar dosyanın bulunduğunu bu parametreye bakarak anlayabilir.

- İşlev çok fazla dosya bulursa ve heap alanı bu dosyaların hepsini tutacak kadar büyük değilse ya da hiç dosya bulunamazsa *NULL* adresine geri dönmelidir. İşlev *NULL* adresi ile geri döndüğünde ikinci parametreyle alınan adrese yerleştirilecek sayı bellek yetersizliği durumunda koşulu sağlayan tüm dosyaların sayısı, dosya bulunmaması durumunda ise sıfır olmalı.

- Karşılaştırma işlevi sıfır dışı bir değerle dönerse bulunan dosya diziye yerleştirilmeli, sıfır ile geri dönerse diziye yerleştirilmemeli.

- Yol ifadesi *.* içermemeli yalnızca dizin belirtilmeli. Dizin isminin sonunda \ bulunabilir.

***struct ffblk* yapısı *DOS*'un *findfirst* ve *findnext* işlevleri tarafından kullanılmaktadır. *Microsoft Windows* derleyicilerinde bu yapı yerine *_finddata_t* yapısı kullanılmaktadır. Yapıların hangi elemanlara sahip olduğunu ilgili kaynaklardan elde edebilirisiniz.**

C:\WINDOWS dizininde *100,000 byte*'tan daha büyük olan dosyaları bulmak isteyelim:

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

struct ffblk *scandir(const char  *path, int *pcount, int (*fcmp)(const
struct ffblk *));

int cmp(const struct ffblk *pFile)
{
    return pFile->ff_fsize > 100000L;
}

int main(void)
{
    struct ffblk *pInfo;
    int count, i;

    pInfo = scandir("C:\\WINDOWS", &count, cmp);
    if (pInfo == NULL) {
        if (count == 0)
            printf("Koşulu sağlayan dosya yok!..\n");
        else
            printf("Bellek yetersiz: %d tane dosya var!..\n", count);
        exit(1);
    }
```

```
    for (i = 0; i < count; ++i)
        printf("%s %ld\n", pInfo[i].ff_name, pInfo[i].ff_fsize);
    free(pInfo);

    return 0;
}
```

****

139. DİZİN AĞACININ DOLAŞILMASINA İLİŞKİN ÇALIŞMALAR (1)

Bir dosyayı o anda bulunulan sürücünün dizin ağacı içinde arayan, bulduğunda o dizine geçişi sağlayan *fcdir* isimli programı yazın.
Programın kullanım biçimi şöyle olmalı:

```
fcdir [seçenek] <dosya ismi>
```

Seçenek için - ya da / karakterleri kullanılabilir.

Seçenekler:
/n: n bir sayıdır. Eğer dosyadan birden çok varsa n'inci bulduğu dizine geçer.
/s: Bu seçenekte dosya her bulunduğunda dizine geçip geçilmeyeceği sorulur. Örneğin:

```
C:\> fcdir -s den.c

C:\PROJECT\LIB\C\SCR\DOS\OLD (Y/N) : N
C:\PROJECT\LIB\C\NDP (Y/N) : N
C:\PROJECT\LIB\C\MATRIX\GENERAL (Y/N) : N
C:\PROJECT\LIB\C\ASSERT (Y/N) : Y
C:\PROJECT\LIB\ASSERT>
```

/h: Yardım. Programın kullanım bilgisini gösterir. Yalnızca program ismi yazıldığında da aynı yardım bilgisi çıkmalıdır.

```
C:\> fcdir

Usage: FCDIR [options] searchfile
-n<number>              Search file depth of the same file
-s                      Step by step conformation mode
-h                      Help
```

Program hiçbir seçenek belirtilmeden de çalıştırılabilir. Bu durumda dosyanın ilk bulunduğu dizine geçilir.
Dizine geçmek için, bildirimi *Borland DOS* derleyicilerinde "*dir.h*" başlık dosyası içinde yapılmış olan *chdir* işlevini, *Microsoft Windows* derleyicilerinde bildirimi *"direct.h"* başlık dosyası içinde bulunan *_chdir* işlevini, nihayet *POSIX* sistemlerinde bildirimi *unistd.h* başlık dosyası içinde yapılan *chdir* işlevini çağırabilirsiniz:

```
int chdir (const char *path);
```

İşlev parametresi ile belirtilen dizine geçişi sağlamalı. İşlev başarılı olursa sıfır değerine başarısız olursa *-1* değerine geri dönmeli.
Eğer programı yazabilirseniz aynı programı joker karakterlerini de içerecek biçimde genişletebilirsiniz. Örneğin:

```
fcdir -s *.C
```

****

140. DİZİN AĞACININ DOLAŞILMASINA İLİŞKİN ÇALIŞMALAR (2)

Bir dosyayı tüm dizinlerden silen "*delfile*" isimli programı yazın.
Programın kullanımı aşağıdaki gibidir:

```
delfile dosya_ismi
```

Program tüm dizinleri dolaşarak belirtilen dosyayı silmeli. Dosya ismi * ve ? işareti gibi joker karakterlerini de içerebilir. Dosyanın silinmesi, bildirimi *stdio.h* başlık dosyası içinde bildirilen standart *remove* işleviyle yapılabilir:

```
int remove(const char *fname);
```

****

141. DİZİN AĞACININ DOLAŞILMASINA İLİŞKİN ÇALIŞMALAR (3)

Bir dizini özyinelemeli bir biçimde ağaç olarak silen programı trimdir ismiyle yazın. Program, dizin ismini komut satırı argümanı biçiminde alarak alt dizinlerle birlikte dizini silmeli. Bir dosyayı silmek için standart *remove* işlevini, boş bir dizini silmek için ise *rmdir* işlevini çağırabilirsiniz. Örneğin:

```
trimdir c:\temp
```

ile *temp* dizini alt dizinleriyle birlikte silinmeli.

****

142. DİZİN AĞACININ DOLAŞILMASINA İLİŞKİN ÇALIŞMALAR (4)

Bir *DOS* ya da *POSIX* komutunu bir dizinden başlayarak onun her alt dizinine geçerek çalıştıran aşağıdaki işlevi yazın:

```
int process_cmd(const char *dirpath, const char *command);
```

İşlevin birinci parametresi başlangıç dizini, ikinci iparametresi çalıştırılacak *DOS* komutudur. İşlevin geri dönüş değeri herhangi bir hata durumunda *0*, başarı durumunda ise sıfır dışı bir değerdir.
Her alt dizin bulunduğunda önce o alt dizine geçilmeli, sonra da standart *system* işleviyle komut uygulanmalıdır. *system* işlevi, belirtilen bir *shell* komutunu çalıştırır

```
/*  Dosyaları */

int process_cmd(const char *dirpath, const char *command);

int main(void)
{
    if (!process_cmd("c:\\", "del *.bak")) {
        fprintf(stderr, "Command execute error!..\n");
        exit(EXIT_FAILURE);
    }
    printf("Success..\n");

    return 0;
}
```