Ex: 009

#include <stdio.h>

//Crie a função recursiva par(n), que determina se o natural n é par,

//usando apenas subtração.

int par(int n){

    if(n==0){

        return 1;

    }

    else if(n==1){

        return 0;

    }

    else{

        par(n-2);

    }

}

int main(){

    int n;

    printf("Digite um numero: ");

    scanf("%d", &n);

    if(par(n)!=1){

        printf("%d não é par",n);

    }

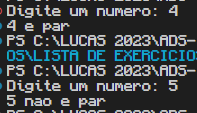
    else{

        printf("%d é par",n);

    }

    return 0;

}



Ex: 010

#include <stdio.h>

//Crie a função recursiva prod(m,n),

//que devolve o produto

//de dois números naturais m e n,

//usando apenas soma e subtração

int prod(int m, int n){

    if(n==1){

        return m;

    }

    return m + prod(m, n-1);

}

int main(){

    int m,n;

    printf("Digite dois numeros: ");

    scanf("%d %d", &m, &n);

    printf("%d x %d = %d", m, n, prod(m,n));

    return 0;

}



Ex: 011

//Crie a função recursiva quoc(m,n), que devolve o quociente da divisão inteira do número natural m pelo número natural n!=0, usando apenas soma e subtração.

#include <stdio.h>

int quoc(int m, int n){

    if(n==0){

        return 0;

    }

    if(n<=m){

        return  1 + quoc(m-n,n);

    }

    else{

        return 0;

    }

}

int main(){

    int m,n;

    printf("Digite dois numeros: ");

    scanf("%d %d", &m, &n);

    printf("%d / %d = %d", m, n, quoc(m,n));

    return 0;

}



Ex: 012

// Crie a função recursiva resto(m,n), que devolve o resto da divisão inteira do número natural m pelo número natural n≠0, usando apenas subtração.

#include <stdio.h>

int resto(int m, int n)

{

    if (n == 0)

    {

        return 0;

    }

    else if (m - n < n)

    {

        return m - n;

    }

    else

    {

        return resto(m - n, n);

    }

}

int main()

{

    int m, n;

    printf("Digite dois numeros: ");

    scanf("%d %d", &m, &n);

    printf("%d %% %d = %d", m, n, resto(m, n));

    return 0;

}



Ex: 013

//O quadrado de um natural n é a soma dos n primeiros ímpares, i.e., n2 = 1 + 3 + 5 + ⋅⋅⋅ + (2n−1). Crie a função recursiva q(n), que devolve o quadrado de n, como base nesta informação.

#include <stdio.h>

int q(n){

    if(n==1){

        return 1;

    }

    else{

        return (2\*n-1) + q(n-1);

    }

}

int main(){

    int n;

    printf("Digite um numero: ");

    scanf("%d", &n);

    printf("%d^2 = %d", n, q(n));

    return 0;

}



Ex: 014

#include <stdio.h>

#include <string.h>

//Crie a função recursiva sd(n), que devolve a soma dos dígitos do número natural n. Por exem-plo, a chamada sd(7859) deve devolver 29 (pois 7+8+5+9 = 29).

// n[0] n[1] n[2] n[3]

// 7    8    5    9

int charToInt(int n){

    switch (n)

    {

    case 48:

        return 0;

        break;

    case 49:

        return 1;

        break;

    case 50:

        return 2;

        break;

    case 51:

        return 3;

        break;

    case 52:

        return 4;

        break;

    case 53:

        return 5;

        break;

    case 54:

        return 6;

        break;

    case 55:

        return 7;

        break;

    case 56:

        return 8;

        break;

    case 57:

        return 9;

        break;

    default:

        break;

    }

}

int sd(int \*n, int l)

{

    if(l==0){

        return 0;

    }

    else{

        return n[l-1] + sd(n, l-1);     //somar um a menos no começo pois a string começa em n+1

    }

}

int main()

{

    int n[100];

    char s[100];

    printf("Digite um numero: ");

    scanf("%s", s);

    for(int i=0; i<strlen(s); i++){ //converter char em int e converter codigo int em int

        n[i] = (int) s[i];

        n[i] = charToInt(n[i]);

        //printf("%d\n", n[i]);

    }

    int l = strlen(s);              //passar tamanho total da string

    printf("soma dos digitos = %d\n", sd(n,l));

    return 0;

}



Ex: 015

//Crie a funÃ§Ã£o recursiva qd(n), que devolve a quantidade de dÃ­gitos binÃ¡rios para representar o natural n. Por exemplo, a chamada qd(13) deve devolver 4 (pois 13 em binÃ¡rio Ã© 1101).

/\*

Passos de conversÃ£o:

Divida o nÃºmero por 2.

Obtenha o quociente inteiro para a prÃ³xima iteraÃ§Ã£o.

Obtenha o resto do dÃ­gito binÃ¡rio.

Repita as etapas atÃ© que o quociente seja igual a 0.

\*/

#include <stdio.h>

int qd(int n){

    if(n==0){

        return 0;

    }

    else{

        return 1+qd(n/2);

    }

}

int main(){

    int n;

    printf("Digite um numero: ");

    scanf("%d", &n);

    printf("O numero %d em binario tem %d digitos", n, qd(n));

    return 0;

}



Ex: 016

//Crie a funÃ§Ã£o recursiva h(n), que devolve o nÃºmero mÃ­nimo de movimentos para resolver o problema das Torres de HanÃ³i com n discos. Por exemplo, h(3) deve devolver 7

#include <stdio.h>

#include <math.h>

int h(int n){

    if(n==1){

        return 1;

    }

    else{

        return 2\*(h(n-1))+1;

    }

}

int main(){

    int n;

    printf("Digite um numero: ");

    scanf("%d", &n);

    printf("O numero minimo para resolver uma torre de hanoi de %d discos eh %d", n, h(n));

    return 0;

}



Ex: 017

//Crie a funÃ§Ã£o recursiva inv(s,p,u), que inverte a string s, cujo primeiro caractere estÃ¡ na posiÃ§Ã£o p e cujo Ãºltimo caractere estÃ¡ na posiÃ§Ã£o u. A funÃ§Ã£o deve devolver s como resposta

//abacate 0 6

//ebacata 1 5

//etacaba 2 4

//etacaba 3 3

#include <stdio.h>

#include <string.h>

void inv(char \*s, int p, int u){

    if(p!=u){

        int aux;

        aux=s[p];

        s[p]=s[u];

        s[u]=aux;

        inv(s,p+1,u-1);

    }

}

int main(){

    char s[100];

    int p,u;

    printf("Digite uma string: ");

    fgets(s,100,stdin);

    printf("Digite a posicao inicial e final: ");

    scanf("%d %d", &p, &u);

    inv(s,p,u);

    printf("A string invertida eh: \n");

    printf("%s", s);

    return 0;

}



Ex: 018

//Crie a funÃ§Ã£o recursiva pal(s,p,u), que informa se a string s, cujo primeiro caractere estÃ¡ na posiÃ§Ã£o p e cujo Ãºltimo caractere estÃ¡ na posiÃ§Ã£o u, Ã© palÃ­ndroma (ignorando brancos).

#include <stdio.h>

//subi no onibus  0 13

// ubi no onibu   1 12

//  bi no onib    2 11

//   i no oni     3 10

//     no on      4 9

//     no on      5 9

//      o o       6 8

//                7 7

int pal(char \*s, int p, int u){

    if(s[p]==s[u]){

        if(p==u && s[p]==s[u]){

            return 1;

        }

        return pal(s,p+1,u-1);

    }

    else if(s[p]==' '){

        return pal(s,p+1,u);

    }

    else if(s[u]==' '){

        return pal(s,p,u-1);

    }

    else{

        return 0;

    }

}

//subinoonibus

int main(){

    char s[100];

    int p,u;

    printf("Digite uma string: ");

    fgets(s,100,stdin);

    printf("Digite a posicao inicial e final: ");

    scanf("%d %d", &p, &u);

    if(pal(s,p,u)==1){

        printf("A cadeia %s eh palindroma", s);

    }

    else{

        printf("A cadeia %s nao eh palindroma", s);

    }

    return 0;

}

