 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	

1. Considere as seguintes funções *minusculoParaMaiusculo* e *maiusculoParaMinusculo*, que se apresentam em seguida. A primeira tem como objetivo converter um carácter minúsculo (que recebe como parâmetro) e devolver a sua conversão em maiúsculo e a segunda função implementa o comportamento inverso. Em ambas as funções, no caso do carácter de entrada não se encontrar no alfabeto minúsculo (ou maiúsculo), o carácter de saída será igual ao de entrada.

Considere a tabela com os caracteres ASCII e a sua numeração decimal que está em anexo e efetue as alterações necessárias às funções apresentadas para que estas tenham o comportamento desejado.

```
char minusculoParaMaiusculo(char car){
    char res;


    res = car;
    if ((car >= 'a') || (car <= 'z')){
        res = car + 'a' + 'A';
    }

    return res;
}

char maiusculoParaMinusculo(char car){
    char res;

    res = car;
    if ((car >= 'Z') && (car <= 'A')){
        res = car - 'A' + 'a';
    }
    return res;
}
```

2. Implemente a função *multimpares* que recebe como parâmetros um vetor de inteiros *seqInt* e o seu tamanho *n*. A função deve solicitar ao utilizador *n* números inteiros entre 0 e 100 e deve guarda-los pela ordem inversa no vetor *seqInt*. A função deve ainda devolver o produto dos números ímpares que o utilizador introduzir.
3. Considere o programa que se apresenta de seguida. Qual o seu output na consola?

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	

```

void func(int v[], int tam){
    int i, j, x, min;

    for(i=0; i< tam; i++){
        min=i;

        for(j = i+1; j < tam; j++){

            if(v[j] < v[min]){
                min =j;
            }
        }
        x = v[i];
        v[i]=v[min];
        v[min]=x;
    }

    for(i=0; i < tam; i++){
        printf("%d \n", v[i]);
    }
}

int main(){

    int v[6]={3,4, 1,5,6,2};

    func(v,6);

    return 0;
}

```


4. Uma função polinomial é uma função que pode ser expressa da seguinte forma:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0 x^0 = \sum_{i=0}^n a_i x^i,$$

em que n é um número inteiro não negativo que identifica o grau do polinómio, e os números a_0, a_1, \dots, a_n são constantes, chamadas coeficientes do polinómio. Um polinómio pode ainda ser representado como um array de inteiros de tamanho $n+1$, que tem em cada posição um coeficiente.

Por exemplo, o polinómio (de grau 3) $P(x) = 2x^3 - 4x^2 + 5x^1 + 13x^0$ pode ser representado como:

2	-4	5	13
---	----	---	----

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	

É ainda possível calcular o valor de um polinómio num dado ponto x . Por exemplo, o valor do polinómio anterior para $x = 2$ é dado por:

$$P(2) = 2 * 2^3 - 4 * 2^2 + 5 * 2^1 + 13 * 2^0 = 16 - 16 + 10 + 13 = 23$$

Implemente a função *calculaPolinomio*, cuja assinatura é dada de seguida, que calcula o valor de um polinómio num dado x , dados um array com os seus coeficientes, o seu grau e o valor de x .

```
/**
 * Calcula o valor de um polinómio, num determinado ponto.
 *
 * @param coefs o array com os coeficientes do polinómio
 * @param grau o grau do polinómio
 * @param x ponto para o qual se pretende calcular o valor
 *
 * @return o valor do polinómio
 */
int calculaPolinomio(const short coefs[], const unsigned short grau,
                    const unsigned int x);
```


5. Implemente a função *contaPalavras*, cuja assinatura é dada de seguida, que conta quantas palavras com n ou mais caracteres existem num array de caracteres *str*.

```
/**
 * Conta quantas palavras existem na string str com n ou mais caracteres.
 * Assume que cada duas palavras são separadas por um espaço e que não existe
 * pontuação.
 *
 * @param str o array de caracteres no qual pesquisar
 * @param n o tamanho mínimo das palavras a pesquisar
 *
 * @return o número de palavras com n ou mais caracteres
 */
int contaPalavras(const char str[], unsigned short n);
```

6. Considere a função *estimar_pi*, que se apresenta de seguida. Esta função tem como objetivo calcular uma aproximação do número pi utilizando a série de Gregory-Leibniz. Esta série pode ser representada da seguinte forma:

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \dots$$

A função *estimar_pi* tem como parâmetro um inteiro n , que indica o número de pares de operações que se pretende calcular para estimar π . A tabela abaixo mostra o resultado esperado para vários valores de n .

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	

<i>N</i>	<i>Resultado</i>
1	$\pi = \frac{4}{1} - \frac{4}{3} = 2.666667$
2	$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} = 2.895238$
3	$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} = 2.976046$

Efetue as alterações necessárias à função *estimar_pi* para que esta tenha o comportamento desejado.

```
float estimar_pi(int n) {
    int i = 0, j = 0;
    float res = 0;

    while(i < n) {
        res = res + 4.0 / j;
        j = j + 2;
        res = res + 4.0 / j;
    }

    return res;
}
```


- Implemente a função *soma_pares* que pede ao utilizador dois números inteiros *n* e *lim* entre 0 e 1000. De seguida, deve pedir ao utilizador *n* números inteiros entre 0 e *lim* e devolver o somatório dos números pares que o utilizador introduzir.
- Implemente a função *trocaChar* que, dado um array de caracteres, dois caracteres *a* e *b* e dois inteiros *i* e *f*, troca todas as ocorrências de *a* no array, entre as posições *i* e *f*, por *b*. A função deve devolver o número de trocas efetuadas.
- Considere a função *estimar_e*, que se apresenta em seguida. Esta função tem como objetivo calcular uma aproximação da constante *e* (base do logaritmo natural ou constante de Napier) utilizando a seguinte série infinita ("!" denota a função *factorial*⁽¹⁾):

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

Desenvolvendo os primeiros 6 termos da série vem:

$$e = \frac{1}{1} + \frac{1}{1} + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} \dots$$

A função *estimar_e* tem como parâmetro um inteiro *k*, que indica o número máximo de iterações (*k* inicial é zero) que se pretende executar para estimar *e*. A tabela abaixo mostra o resultado esperado para vários valores de *k*:

 <div>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</div>	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores		Hora
	Unidade Curricular Fundamentos de Programação		Duração

k	Resultado
0	$e = \frac{1}{1} = 1$
1	$e = \frac{1}{1} + \frac{1}{1} = 2$
2	$e = \frac{1}{1} + \frac{1}{1} + \frac{1}{2} = 2.5$
3	$e = \frac{1}{1} + \frac{1}{1} + \frac{1}{2} + \frac{1}{6} = 2.666666(6)$

Efetue as alterações necessárias à função *estimar_e* para que esta tenha o comportamento desejado.

Relembra-se que $e = 2.718281828459045235360287471352662497757...$

⁽¹⁾ Considere a função definida em C por: `double factorial(int);`

```
float estimar_e (int k) {
    int i, j=0;
    float res = 0;
    for (i=2; i>=k; i++)
        res = res + 1 / factorial(j);
    return res;
}
```


- Implemente a função *produto_divisiveis_3* que pede ao utilizador dois números inteiros n e lim entre 0 e 100. De seguida, deve pedir ao utilizador n números inteiros cada um entre 0 e lim e devolver o produto (\prod) dos números divisíveis por 3 que o utilizador introduziu.
- Implemente a função *removerBrancoStr* que, dada uma string, remova eventuais caracteres brancos (' ') no final da string. A função deve devolver o número de caracteres brancos removidos.
- Considere a função *estimar_G*, que se apresenta de seguida. Esta função tem como objetivo calcular uma aproximação da constante de Catalan utilizando a seguinte série infinita:

$$G = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2}$$

Desenvolvendo os primeiros 6 termos da série vem:

$$G = 1 - \frac{1}{3^2} + \frac{1}{5^2} - \frac{1}{7^2} + \frac{1}{9^2} - \frac{1}{11^2} \dots$$

A função *estimar_G* tem como parâmetro um inteiro n , que indica o número de termos a calcular para estimar G . A tabela abaixo mostra o resultado esperado para vários valores de n .

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	

n	Resultado
0	$G = 1$
1	$G = 1 - \frac{1}{9} = 0.88888(8)$
2	$G = 1 - \frac{1}{9} + \frac{1}{25} = 0.92888(8)$
3	$G = 1 - \frac{1}{9} + \frac{1}{25} - \frac{1}{49} = 0.90848$

Efetue as alterações necessárias à função *estimar_G* para que esta tenha o comportamento desejado.

```
float estimar_G (int n) {
    int i;
    float res=0, div;

    for (i=1; i <= n; i++) {
        div = (2 * i + 1)2;
        if (i mod 2 == 0)
            res += 1/div;
        else
            res -= 1/div;
    }
    return res;
}
```


13. Implemente a função *e_perfeito* que recebe como argumento um número inteiro e determina se este é um número perfeito ou não. Um número perfeito é aquele cuja soma dos seus divisores (excluindo o próprio) é igual ao próprio número.

Exemplos:

6 = 1 + 2 + 3 – é perfeito

12 = 1 + 2 + 3 + 4 + 6 – não é perfeito

14. Implemente a função *contaChar* que, dada uma string e um carácter, conta quantas ocorrências desse carácter existem na string. A função deve devolver o número de ocorrências do carácter.
15. Considere a função *fibonacci*, que se apresenta de seguida. Esta função tem como objetivo calcular o *n*ésimo número da sequência de fibonacci. Lembre-se que, nesta sequência, os primeiros 2 elementos são 0 e 1 e cada número subsequente pode ser obtido pela soma dos dois números anteriores (e.g. 0, 1, 1, 2, 3, 5, 8...). Assim, por exemplo, para $n = 5$ a função deve devolver o valor 3. Para $n < 1$ a função deve devolver o valor -1. Corrija a função de forma a que esta tenha o comportamento desejado.

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	

```

int fibonaci(int n){
    int res, ant1 = 0, ant2 = 1, temp;

    if (n < 1)
        res = -1;
    else if (n == 1)
        res = 0;
    else if (n == 2)
        res = 1;
    else {
        while(n > 0){
            temp = ant1 - ant2;
            ant1 = ant2;
            ant2 = temp;
            n++;
        }
    }

    return res;
}

```

16. Escreva uma função que leia do utilizador um número indeterminado de inteiros entre 0 e 1000 e, no final, devolva a multiplicação do maior pelo menor número introduzido. A leitura de números deve terminar quando o utilizador introduzir o número -1. A função deve retornar -1 se nenhum número positivo for introduzido.
17. Considere o programa que se apresenta de seguida. Qual o seu output na consola?

```

int contaIteracoes(int lista[], int tam, int pos){
    int i = pos, conta = 0;

    while(i >= 0 && i < tam){
        i = i + lista[i];
        conta++;
        printf("Iteração %d, Valor de i: %d\n", conta, i);
    }

    return conta;
}


int main(int argc, char** argv) {

    int lista[] = {2,5,-3,1,3,-2};

    printf("Total de Iterações: %d\n", contaIteracoes(lista, 6, 5));
}

```

18. Implemente a função *mediaPositivos* que, dada como parâmetro uma lista de inteiros e o seu tamanho, devolve a média dos seus valores positivos.

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Exercícios Teste 1	Ano letivo 2019/2020	Data
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora	
	Unidade Curricular Fundamentos de Programação	Duração	