Teste, 31 de Maio de 2022

1. Implemente a função `int nesimo(int a[], int N, int i)` que dado um array de tamanho `N > 0` e um inteiro `0 < i <= N` devolve o `i`-ésimo menor elemento do array. Por exemplo, se i == 1 a função deve retornar o menor elemento do array.

2. Implemente a função `LInt removeMaiores(LInt l, int x)` que remove **de uma lista ordenada** `l` todos os elementos maiores que `x`, devolvendo a lista resultante. Considere a definição usual do tipo `LInt`.

```
typedef struct LInt_nodo {
     int valor;
     struct LInt_nodo *prox;
} *LInt;
```

3. Implemente a função `LInt caminho(ABin a, int x)` que, dada uma **árvore binária de procura** `a` e um valor `x`, devolve uma lista com todos os valores desde a raiz até `x` (inclusivé). Se `x` não existir na árvore, deve devolver `NULL`. Considere a definição usual do tipo `ABin` (o tipo `LInt` foi dado na questão anterior).

```
typedef struct ABin_nodo {
     int valor;
     struct ABin_nodo *esq, *dir;
} *ABin;
```

4. Implemente a função `void inc(char s[])` que, dada uma uma string `s` com um número em decimal, incrementa esse número numa unidade. Assuma que a string tem espaço suficiente para armazenar o número resultante. Por exemplo, se a string for `"123"` deverá ser modificada para "124". Se for "199" deverá ser modificada para "200".

5. Implemente a função `int sacos(int p[], int N, int C)` que, dado um array com os pesos de `N` produtos que se pretende comprar num supermercado, e a capacidade `C` dos sacos desse supermercado, determine o número mínimo de sacos necessários para transportar todos os produtos. Por exemplo, se os pesos dos produtos forem `{3,6,2,1,5,7,2,4,1}` e `C == 10`, então bastam `4` sacos. Se os pesos forem `{3,3,3,3,5,5,11}` e `C == 11`, então bastam `3` sacos. Em geral, para descobrir este mínimo teria que testar todas as possíveis maneiras de ensacar os produtos. Se não conseguir implementar essa estratégia óptima, implemente outra que devolva uma aproximação razoável.

Exame, 21 de Junho de 2022

1. Implemente **de forma eficiente** uma função `int pesquisa (int a[], int N, int x)` que, dado um **array ordenado** de tamanho `N > 0`, devolve um índice onde se encontra o valor `x`. Caso `x` não exista no array a função deverá devolver `-1`

2. Implemente uma função `void roda (LInt *l)` que move o último elemento da lista para a cabeça da mesma (sem alocar nova memória). Considere a definição usual do tipo `LInt`.

```
typedef struct LInt_nodo {
     int valor;
     struct LInt_nodo *prox;
} *LInt;
```

3. Implemente uma função `int apaga (ABin a, int n)` que apaga **`n`** nodos de uma árvore binária. O critério para escolha de quais os nodos a apagar é livre. Se a árvore tiver menos do que `n` nodos então a deve apagar todos. A função deve devolver o número de nós efetivamente apagados. Considere a definição usual do tipo `ABin`.

```
typedef struct ABin_nodo {
     int valor;
     struct ABin_nodo *esq, *dir;
} *ABin;
```

4. Implemente uma função `void checksum (char s[])` que, dada uma uma string `s` com um identificador só com dígitos, acrescenta-lhe um dígito de controle no final calculado de acordo com o método de Luhn. Neste método, o dígito de controle a incluir deve fazer com que a soma de todos os dígitos (incluindo o próprio dígito de controle) seja um múltiplo de 10. No entanto, no caso dos dígitos **em posições pares** (a começar do final) o que deve ser somado são os dígitos do número correspondente ao seu dobro. Por exemplo, dado o identificador `"9871"`, a soma em questão corresponde a `9+`**`1+6`**`+7+`**`2`**` = 25` (note como o dígito 1 e 8 foram substituídos, respectivamente, por 2 e 1+6). Como a soma é 25, o dígito de controle a acrescentar deve ser 5, pelo que a string no final deverá ser `"98715"`.

5. Implemente uma função `int escolhe (int N, int valor[], int peso[], int C, int quant[])` cujo objetivo é determinar a quantidade de produtos que um vendedor ambulante deve transportar. O vendedor tem à sua disposição uma quantidade ilimitada de `N` produtos diferentes, cujos valores e pesos estão guardados nos arrays `valor` e `peso`, respectivamente, mas só tem capacidade para

transportar `C` kg. A função deve tentar maximizar o valor total dos produtos a transportar, valor este que deve ser devolvido, e colocar no array `quant` a respectiva quantidade de cada produto. Por exemplo, se tivermos `3` produtos com valores `[20,150,30]` e pesos `[2,10,3]` e capacidade para `14` kg, então uma escolha ideal de quantidades seria `[2,1,0]`, correspondente ao valor total de `190`. Mesmo que não consiga implementar uma estratégia de escolha óptima, implemente outra que ache razoável. O critério mais importante é o peso total não ultrapassar a capacidade de transporte C.

Teste, 20 de Maio de 2023

1. Implemente a função `int perfeito(int x)` que testa se um número inteiro é perfeito, isto é, se é igual à soma dos seus divisores próprios. Por exemplo, 28 é um número perfeito, uma vez que os seus divisores próprios são `1`, `2`, `4`, `7` e `14` (`1+2+4+7+14==28`).

2. Implemente a função `void ordena(Ponto pos[], int N)` que dado um array com `N` pontos ordena esses pontos por ordem crescente da distância à origem. Por exemplo se o array for `{{3,3},{2,1},{-1,0}}` depois de ordenado deverá ficar com o conteúdo `{{-1,0},{2,1},{3,3}}`. O tipo `Ponto` é definido da seguinte forma (note que as coordenadas dos pontos são números inteiros).

```
typedef struct {
     int x,y;
} Ponto;
```

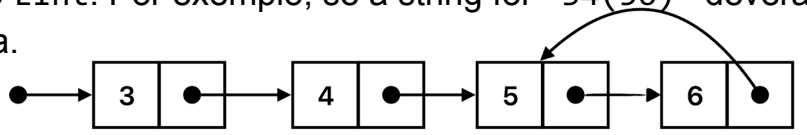
3. Apresente uma definição da função `int depth(ABin a, int x)` que devolve o **menor nível** a que um elemento `x` se encontra na árvore (ou -1 se `x` não se encontra na árvore). Considere a definição usual do tipo `ABin`. Considere ainda que a raiz se encontra no nível 0.

```
typedef struct abin_nodo {
     int valor;
     struct abin_nodo *esq, *dir;
} *ABin;
```

4. Implemente a função `int wordle(char secreta[], char tentativa[])` que dada uma palavra secreta que se pretende descobrir e uma tentativa com o mesmo tamanho devolve o número de caracteres na palavra tentativa em que o utilizador já acertou. Ambas as palavras só contêm letras minúsculas. A função deve também modificar a tentativa substituindo todas as letras que não tem correspondente na palavra secreta por um `'*'` e convertendo para maiúscula as letras que estão na posição certa. Por exemplo se a palavra secreta for `"laranja"` e a tentativa for `"cerejas"` a função deve devolver 1 e alterar a tentativa para `"**R*ja*"` (apenas o `'r'` está na posição certa e os caracteres `'j'` e `'a'` aparecem no segredo noutras posições). Se a tentativa for `"bananas"` a função deve devolver 3 e alterar a tentativa para `"*A*ANa*"`.

5. Implemente a função `LInt periodica(char s[])` que dada uma string com uma sequência infinita periódica de dígitos constrói uma lista (circular) com esses dígitos.

Assuma que a parte da sequência que se repete indefinidamente está representada entre parênteses e aparece sempre no final da string. Assuma também a definição usual do tipo `LInt`. Por exemplo, se a string for `"34(56)"` deverá ser construída a seguinte lista.




```
typedef struct lint_nodo {
     int valor;
     struct lint_nodo *prox;
} *LInt;
```

Exame, 13 de Junho de 2023

1. Implemente a função `int isFib(int x)` que testa se um número `x` pertence à sequência de Fibonacci (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...).

2. Implemente a função `int moda(Aluno turma[], int N)` que, dado um array com a informação de `N` alunos, calcula a nota final mais frequente. Se houver mais do que uma nota final com a frequência máxima, devolva uma delas. Se a nota final mais frequente for "Reprovado" então deve devolver `0`. O teste vale 80% da nota final e os mini-testes 20%.

```
typedef struct {
     float teste, minis;
} Aluno;
```

3. Apresente uma definição **iterativa** da função `int take(int n, LInt *l)` que, dado um inteiro `n` e uma lista ligada de inteiros `l`, apaga de `l` todos os nodos para além do `n`-ésimo (libertando o respectivo espaço). Se a lista tiver `n` ou menos nodos, a função não altera a lista. A função deve devolver o número de nodos apagados.

```
typedef struct lint_nodo {
     int valor;
     struct lint_nodo *prox;
} *LInt;
```

4. Apresente uma definição da função `int verifica(char frase[], int k)` que testa se todas as palavras que ocorrem numa `frase` têm pelo menos `k` caracteres.

5. Implemente a função `ABin reconstroi(char s[])` que dada uma string com uma string com uma travessia pré-order de uma árvore de dígitos, onde os apontadores nulos aparecem marcados com um `'*'`, reconstrói a árvore original. Por exemplo, se a string for `"34**52**5*6**"` deverá ser devolvida a seguinte árvore.

```
typedef struct abin_nodo {
     int valor;
     struct abin_nodo *esq, *dir;
} *ABin;
```