INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Charqueadas

Tecnólogo em Sistemas para Internet
Disciplina de Programação Estruturada – 2 Semestre
André Luís Del Mestre Martins

Lista de exercícios 5 – Alocação dinâmica – **Prazo: combinado em aula**

INSTRUÇÕES:

A – Todos os arquivos compactados em formato ZIP (.zip). Inclua apenas os códigos-fonte (.c e .h), ou seja, não me envie os executáveis (.exe). O zip deve seguir o padrão: *[SEU_NOME].zip.*
B – Todas as funções solicitadas na lista de exercícios devem estar dentro de sua biblioteca. A biblioteca deve seguir o padrão: *[SEU_NOME].h* para os protótipos e *[SEU_NOME].c* para a descrição das funções.
C – Os códigos-fonte dos exercícios devem conter APENAS a função `main()` e devem seguir o padrão: *ex[NUMERO].c.* Lembre-se que são apenas 6 arquivos.
D – A lista de exercícios é entregue pelo Google Classroom.

PARTE 1 – INT e FLOAT

1 – Escreva a função `int * copyArray(int* vet, int tamanho)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será uma cópia do vetor `vet`. `tamanho` é o tamanho de `vet`. Saída esperada:

```
Vetor A = [1,2,3,4,5] Vetor B = NULL.
Copiando A em B...
Vetor A = [1,2,3,4,5] Vetor B = [1,2,3,4,5].
```

2 – Escreva a função `int * subArray(int * vet, int tamanho, int inicio, int fim)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será um subvetor de `vet` com valores entre os índices `inicio` e `fim`. `tamanho` é o tamanho de `vet`. Faça todas as validações necessárias relativas ao intervalo e tamanho de `vet`. Saída esperada:

```
Vetor A = [1,5,3,4,2] Vetor B = NULL.
Gerando subvetor entre os indices 4 e 6...
Vetor A = [1,5,3,4,2] Vetor B = NULL.
Gerando subvetor entre os indices 1 e 3...
Vetor A = [1,5,3,4,2] Vetor B = [5,3,4].
```

3 – Escreva a função `int * catArray(int * vetA, int tamA, int * vetB, int tamB)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será um vetor dinâmico dado pela concatenação de `vetA` e `vetB`. `tamanho` `tamA` e `tamB` são o tamanho dos vetores `vetA` e `vetB`, respectivamente. Saída esperada:

```
Vetor A = [1,5,3,4,2] Vetor B = [777,333,888].
Concatenando vetores A e B...
Vetor C = [1,5,3,4,2,777,333,888]
Concatenando vetores B e A...
Vetor D = [777,333,888,1,5,3,4,2]
```

4 – Escreva a função `int * biggerThanArray(int * vet, int num, int tamanho)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será um subvetor de `vet` com valores maiores que `num`. `tamanho` é o tamanho de `vet`. Saída esperada:

```
Vetor A = [1,5,3,4,2] Vetor B = NULL.
Gerando subvetor com valores acima de 3...
Vetor A = [1,5,3,4,2] Vetor B = [5,4].
```

PARTE 2 – INT e FLOAT com `realloc()`

5 – Escreva a função `int * insertNumBeginArray(int * vet, int num, int tamanho)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será o vetor `vet` mais o número `num` adicionado no seu início. `tamanho` é o tamanho de `vet`. Saída esperada:

```
Vetor A = [1,5,3,4,2] Numero = 666.
Adicionando 666 no inicio do Vetor A...
Vetor A = [666,1,5,3,4,2].
```

6 – Escreva a função `int * insertNumArray(int * vet, int num, int pos, int tamanho)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será o vetor `vet` mais o número `num` adicionado na posição `pos`. `tamanho` é o tamanho de `vet`. Saída esperada:

```
Vetor A = [1,5,3,4,2] Numero = 666.
Adicionando 666 na posicao 3 do Vetor A...
Vetor A = [1,5,3,666,4,2]
```

7 – Escreva a função `int * removeDuplicates(int * vet, int tamanho)` para retornar um ponteiro com um vetor dinâmico de inteiros cujo o conteúdo será o vetor `vet` sem redundância de valores. `tamanho` é o tamanho de `vet`. Saída esperada:

```
Vetor A = [1,5,3,4,2,3,6,5,3,7,0]
Removendo redundancias do Vetor A...
Vetor A = [1,5,3,4,2,6,7,0]
```

PARTE 3 – CHAR e STRING

8 – Escreva a função `char * catString(char*, char*)` para retornar uma string alocada dinamicamente dada pela concatenação de duas strings. Saída esperada:

```
Concatena “Programacao” e “Estruturada”
Resultado: “Programacao Estruturada”
```

9 – Escreva a função `char * copyString(char*)` para retornar a cópia do conteúdo de uma string. Saída esperada:

```
Str1: "Programacao Estruturada", Str2: NULL
Copiando Str1 em Str2...
Str1: "Programacao Estruturada", Str2: "Programacao Estruturada"
```

10 – Escreva a função `char * toggleString(char*)` para retornar uma string formatada da seguinte maneira: a primeira palavra de cada palavra deve ser maiúscula e as demais minúsculas. Saída esperada:

```
Formatando string  "pRogRAMaCao ESTRUTURADA"
Resultado: "Programacao Estruturada"
```

PARTE 4 – CHAR e STRING com `realloc()`

11 – Escreva a função `char * removeOneChar(char*, char)` para remover a primeira ocorrência de um caracter em uma string. Saída esperada:

```
Removendo um ‘r’ de “Programacao Estruturada”
Resultado: “Pogramacao Estruturada”
```

12 – Escreva a função `char * removeCompletelyChar(char*, char)` para remover todas as ocorrências de um caracter em uma string. Saída esperada:

```
Removendo ‘r’ de “Programacao Estruturada”
Resultado: “Pogamacao Estutuada”
```

13 – Escreva a função `char * removeWord(char*, char*)` para remover todas as ocorrências de uma substring em uma string. Saída esperada:

```
Removendo "Progr" de "Programacao Estruturada"
Resultado: "amacao Estruturada"
Removendo "te" de "teste teste teste"
Resultado: "s s s"

Removendo "voce" de "Programacao Estruturada"
Resultado: NULL
```

14 – Escreva a função `char * replaceWord(char*, char*, char*)` para substituir todas as ocorrencias de um caracter por outro caracter em uma string. Saída esperada:

```
Substituindo "Program" por "Super" em "Programacao Estruturada"
Resultado: "Superacao Estruturada"

Substituindo " te" por "-----" de "teste teste teste"
Resultado: "teste-----ste-----ste"

Substituindo "voce" por "outro" em "Programacao Estruturada"
Resultado: NULL
```

8 – *(desenvolvida em aula, mas não foi encapsulada em função)* Escreva a função `int * readArrayFromUser()` para retornar um ponteiro com um vetor dinâmico de inteiros digitado pelo usuário. O programa deverá ter alocado somente o espaço necessário para armazenar os números digitados. Quando o usuário informar o número 0, a função é encerrada. Saída esperada:

```
Diga um numero: 4
Diga um numero: 1
Diga um numero: 4
Diga um numero: 0
Vetor de tamanho 3 lido eh: [4,1,4]
```

Desafio: como você vai retornar o tamanho do vetor dinâmico além do próprio conteúdo vetor?