SCC-502 Algoritmos e Estruturas de Dados 1 **M. Cristina/Jorge**

Lista de Exercícios 1 - Pilhas

A lista de exercícios deverá ser feita individualmente e enviada para o e-mail "jorgehpopae@gmail.com" com o assunto "LISTA_1 - <PRIMEIRONOME>_<NºUSP>", por exemplo, "LISTA_1 – Jorge_9999999". O e-mail deve conter no anexo apenas 1 arquivo compactado, "LISTA1_<NOME>_<NºUSP>.zip", por exemplo, "LISTA1_Jorge_9999999.zip". Neste arquivo, devem estar os exercícios resolvidos em arquivos separados, com nomes "exercicio1.c", "exercicio2.c", etc. A data limite para entrega é **20/09**.

Dado que as funções abaixo estão implementadas em um arquivo "TADPilha.c", resolva os exercícios. Atenção: **a pilha só pode ser acessada por meio das operações definidas no arquivo "TADPilha.c".**

```
#define TAMPILHA 100

typedef struct{

        int topo;

        int itens[TAMPILHA];

} Pilha;


//nao ocorreu erro: flagErro=0;
//erro: flagErro=1;
Pilha* create(int *flagErro);

void push(pilha *p, int valor, int *flagErro);

int pop(pilha *p, int *flagErro);

int empty(pilha *p, int *flagErro);
```

1. Desenvolva um algoritmo para testar se uma pilha P1 tem mais elementos que uma pilha P2. Considere que P1 e P2 já existem.
O protótipo da função deve ser:

```
int testaMaisElementos(pilha *P1, pilha *P2);
//A função retornará 1 para verdadeiro (P1 > P2) e 0 para falso.
```

2. Desenvolva uma operação para transferir elementos de uma pilha P1 para uma pilha P2 (cópia). Siga o protótipo abaixo:

```
void transferirElementos(pilha *P1, pilha *P2, int *flagErro);
//A função retornará 0 em *flagErro para sucesso e 1 para erro
```

3. Desenvolva um algoritmo para inverter a posição dos elementos de uma pilha P. Você pode criar pilhas auxiliares, se necessário. Mas o resultado precisa ser dado na pilha P.

```
void inverter (pilha *P);
```

4. Desenvolva um algoritmo para testar se duas pilhas P1 e P2 são iguais. Duas pilhas são iguais se possuem os mesmos elementos, na mesma ordem.

```
int iguais(pilha *p1, pilha *p2);
//retorna 1 para p1 == p2  e 0 para p1 != p2
```

5. Escreva um programa para verificar se uma expressão matemática tem os parênteses agrupados de forma correta, isto é: (1)se o número de parênteses à esquerda e à direita são iguais e; (2) se todo parêntese aberto é seguido posteriormente por um fechamento de parêntese.
Ex1: As expressões ((A+B) ou (A+B violam a condição 1
Ex2: As expressões)A+B(–C ou (A+B)) – (C + D violam a condição 2

6. Escreva um algoritmo para determinar se uma string de caracteres de entrada é da forma xCy, onde x é uma string consistindo das letras A e B e y é o inverso de x, isto é se x = “ABA”, a string deverá conter os caracteres “ABBCBBA”

7. Suponha uma máquina de calcular que trabalha apenas com números não negativos, e que tem apenas as quatro operações: soma, subtração, produto e divisão inteira. A máquina tem 16 teclas, representadas pelos caracteres:

0 1 2 3 4 5 6 7 8 9 + - * / ^ C E

onde C representa “clear”, e E representa “enter”, que é usada indicar que vai ser fornecido um número. A máquina usa notação Polonesa Reversa, aquela em que o operador vem depois dos operandos.

Escreva um programa que usa uma pilha de inteiros para simular a máquina. Inicialmente, a pilha da máquina está vazia. As ações correspondentes a cada caractere são:

i = 0, … 9	troque o valor x do topo da pilha por x*10 + i
E	empilhe um 0
op = +, -, *, /, ^	tire dois elementos y e x do topo da pilha e empilhe x op y
C	Imprime o resultado na tela e esvazia a pilha

Por exemplo, se o usuário tivesse digitado
E 9 0 E 1 5 E 3 0 + - C

Comando	Pilha
//Comeco do programa – pilha vazia	[ ]

E	[0]
9	[9]
0	[90]
E	[90 0]
1	[90 1]
5	[90 15]
E	[90 15 0]
3	[90 15 3]
0	[90 15 30]
+	[90 45]
-	[45]
C	[ ]