ATIVIDADE AV2 – ALGORITMOS E ESTRUTURAS DE DADOS

1. Analise as premissas abaixo sobre a utilização de vetores e/ou ponteiros.
I - Considerando um vetor (*char vet[10];*), um ponteiro para o vetor (*char *p;*) é dado por *p=&vet[0];*.
II - Uma string é uma cadeia de caracteres com um terminador (*\0*).
III - Considerando um vetor (*char vet[10];*), um ponteiro para o vetor (*char *p;*) é dado por *p=vet;*
Está correto o que se afirma em:

a) I, II e III.
b) I e III, apenas.
c) I e II, apenas.
d) I, apenas.
e) III, apenas.

2. As funções de alocação dinâmica de memória são importantes para a linguagem de programação C. Desta forma, assinale a opção que contém a biblioteca necessária para usar as funções malloc() e free().

a) <ctype.h>.
b) <stdlib.h>.
c) <stdio.h>.
d) <time.h>.
e) <math.h>

3. Analise as afirmações a seguir a respeito de pilhas: I - Novos elementos entram, no conjunto, exclusivamente, no topo da pilha. II - O único elemento que pode sair da pilha em um dado momento, é o elemento do topo. III - as Pilhas são conhecidas como LIFO (*last in, first out*), isto é, o último a entrar é o último a sair. Estão corretas as afirmações?

4. Considere o código abaixo

```
int iVar = 15;
int jVar,*pPont,*qPont;
pPont = &iVar;
jVar = 2 * (*pPont);
qPont = 2 + (pPont);
```

Supõe-se que as posições (endereços) de memória ocupadas pelas variáveis iVar, jVar, pPont e qPont sejam respectivamente 3080, 3084, 3088 e 3096. Responda:

a) Qual será o valor que a pPont assume?
b) O que de fato ocorre, ou seja, qual será o valor que *pPont assume?
c) Qual será o valor de jVar depois da atribuição. Por quê?
d) Qual será o valor de qPont depois da atribuição. Por quê?

5. Qual será a saída deste programa, supondo que i ocupa o endereço 2048 na memória?

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
      int i=5, *p;
      p = &i;
      printf("%p - %d - %d\n", p, *p+2, 3*(*p));
}
```

6. Considerando que p seja uma pilha de 4 posições. Qual será o estado final da pilha após estas operações?

p

```
p = cria_pilha()
empilha (p, 15)
empilha (p, 25)
empilha (p, 30)
desempilha (p)
empilha (p,50)
desempilha (p)
empilha (p,40)
```

Com base no código apresentado na página A SEGUIR (que apresenta uma Pilha de 20 elementos do tipo inteiro utilizando struct, bem como 3 funções: empilha, desempilha e imprimePilha), responda:

7. 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);
```

8. Desenvolva um algoritmo para inverter a posição dos elementos de uma pilha P1. Você pode criar pilhas auxiliares, se necessário. Mas o resultado precisa ser dado na pilha P. Siga o protótipo abaixo:

```
void inverter (pilha *P1);
```

9. 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.

```
void iguais(pilha *P1, pilha *P2);
```

10. Desenvolva um algoritmo para informar o usuário final se uma pilha P1 está cheia. Siga o protótipo abaixo:

```
void cheia(pilha *P1);
```

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO_PILHA 20
typedef struct{
	int vetor[TAMANHO_PILHA]; //tamanho da pilha
	int topo;
}Pilha;

//prototipo da função empilha
void empilha(int valor,Pilha *P1){
	//pilha->topo significa: ponteiro "pilha" apontando para CONTEÚDO de um item de uma struct
	if(P1->topo < TAMANHO_PILHA){ //verificando se a pilha não está cheia
	//daí pode empilhar
	P1->vetor[P1->topo]=valor;
	P1->topo++;
	printf("O valor %d foi adicionado \n",valor);
	}else{
	printf("Nao ha mais espaco na pilha, \n");
	}
}
void desempilha(Pilha *P1){
	if(P1->topo > 0){
	P1->topo--; //desempilha
	printf("Elemento retirado: %d. \n",P1->vetor[P1->topo]);
	}else{
	printf("A pilha está vazia. \n"); //pilha vazia
	}
}
void imprimePilha(Pilha *P1){
	int i;
	printf("\nSegue Impressao da Pilha: \n");
	for(i=((P1->topo)-1);i>=0;i--){ //valor inicial de i é a ultima posição da pilha e daí decrementa
	printf("\t %d \n",P1->vetor[i]);
	}
}

int main() {
	//DECLARA UMA PILHA
	Pilha P1;
	P1.topo=0; // o topo da pilha deve começar em zero
}
```