Universidade Federal do Piauí – UFPI
Campus Senador Helvídio Nunes de Barros – CSHNB
Curso de Sistemas de Informação Bloco: IV
Disciplina: Estruturas de Dados II
Professora: Juliana Oliveira de Carvalho
Acadêmico:.....Matrícula:.....

ATIVIDADE DE FIXAÇÃO 1

1)

```
void altera1(int **p, int *a)
{ **p = *a;
  *a = *a + 100;
}

void altera2(int **p, int *b)
{ *p = b;
  *b = *b +50;
}

void altera3(int *p, int *b)
{ p = b;
  *b = *b +100;
printf("Altera3: b = %d, End. b = %p, p = %p = %d \n",*b,b, p);
}

int main()
{
 int x,y,z, *px, *py, *pz;

 x = 10;
 y = x + 20;

 px = &x;
 py = &y;
 z = *py + 50;
 pz = &z;

printf("x = %d, End. x = %p, px = %p = %d \n",x,&x, px);
printf("y = %d, End. y = %p, py = %p = %d \n",y,&y, py);
printf("z = %d, End. z = %p, pz = %p = %d \n",z,&z, pz);

altera1(&px, &y);
printf("x = %d, End. x = %p, px = %p = %d \n",x,&x, px);
printf("y = %d, End. y = %p, py = %p = %d \n",y,&y, py);
printf("z = %d, End. z = %p, pz = %p = %d \n",z,&z, pz);
getchar();

altera2(&pz, &x);
printf("x = %d, End. x = %p, px = %p = %d \n",x,&x, px);
printf("y = %d, End. y = %p, py = %p = %d \n",y,&y, py);
printf("z = %d, End. z = %p, pz = %p = %d \n",z,&z, pz);
getchar();

altera3(px, &x);
printf("x = %d, End. x = %p, px = %p = %d \n",x,&x, px);
getchar();

return(0);
}
```

Dado o código acima, responda as seguintes questões:

(a) Qual a diferença entre px e x?
(b) Qual a diferença entre px, py e pz?
(c) Quais são os valores impressos pelos primeiros três printfs?
(d) O que muda em relação a primeira sequência de printfs para a segunda?
(e) Quais os valores impressos pela terceira sequência de printfs?
(f) O que muda da segunda sequência para a terceira sequência de printfs?
(g) Quais valores são apresentados na tela na execução de altera3 e por quê?
(h) Quais os valores são apresentados depois da chamada da função altera3? Compare e justifique as diferenças referente ao respondido no item anterior
(i) Explique a diferença entre o altera1, altera2 e altera3.

2) Faça uma função que dado um número inteiro devolva os divisores do mesmo, a função deve ser recursiva.

3) Faça uma função em C que leia um vetor de strings não ordenado e depois faça:
   (a) uma função recursiva com pendência que devolva a string de maior tamanho.
   (b) uma função recursiva sem pendência que devolva a quantidade de strings que iniciam com vogal.
   (c) uma função recursiva que devolva um vetor contendo somente as strings com tamanho >=4 e que iniciam com letra maiúscula.

4) Faça um algoritmo em C que leia 4 números inteiros e então faça:
   (a) Uma função recursiva sem pendência que calcule Máximo Divisor Comum (MDC) entre os números lidos.
   (b) Repita o exercício da letra a, mas agora faça uma função recursiva que deixa pendência.

5) Dado uma lista simplesmente encadeada e ordenada de pessoas:

```
struct pessoa{
        int Id;
        String Nome;
        int AnoNascimento;
        int Altura;
       };

struct lista{
        Struct pessoa Info;
        Struct lista *Prox;
      };
```

Faça as seguintes funções:

(a) Uma função recursiva que devolva o nome da pessoa mais velha da lista, caso exista mais de uma devolva um vetor com os nomes.
(b) Uma função recursiva que devolva a menor e a maior altura.
(c) Uma função recursiva que devolva o número de pessoas com a altura mediana, utilize o resultado do item (b) para calcular a altura mediana.
(d) Uma função recursiva que devolva o número de pessoas abaixo de 40 anos com a altura maior do que a altura mediana.