INSTITUTO FEDERAL DE GOIÁS
CÂMPUS GOIÂNIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
TEORIA GERAL DA ADMINISTRAÇÃO

Nome do Aluno: Luiz Antônio Rodrigues dos Santos
Data: 07/10/2017
Prof. Renan Rodrigues

Estruturas de Dados I - Revisão

1.Seja o gerador da sequência dada por:

•S(1) = 1
•S(n) = S(n-1) + 3, para n ≥ 2.

a) Explique o comportamento desta série.
R: Para n = 2 o resultado é 4, e a cada número incrementado à n maior que 2, S(n) aumenta em um o resultado, ou seja, n = 3 o resultado será 5 e assim por diante.

b) Através da relação de recorrência do enunciado da questão, apresente o valor dos cinco primeiros números da série.
R: 4, 5, 6, 7, 8.

c) Implemente uma função recursiva na linguagem C que deve receber a posição de um elemento da série e retorne o seu respectivo valor.
R:

```
int sequencia_recursiva(int n){
    int seq;

    if(n == 0){
        return 0;
    }else{
        sequencia_recursiva(n - 1);
        seq = (n - 1) + 3;
    }
    return seq;
}

int main(){
    int pos, res;

    printf("Digite a posicao do elemento: ");
    scanf("%d", &pos);

    res = sequencia_recursiva(pos + 1);
    printf("Elemento %d tem valor: %d\n\n", pos, res);
}
```

d) Considere a existência de uma implementação não-recursiva. Neste caso, provavelmente, qual implementação é mais eficiente: a versão recursiva ou uma versão não-recursiva? Justifique.
R: No meu ponto de vista a implementação com a função recursiva e não recursiva são quase equivalentes, sendo que, a função não recursiva o código ficou com 7 linhas e gastou uma variável a mais, e o código da função recursiva o código ficou com 10 linhas, ainda sim a implemetação recursiva compilou em menos tempo, então a versão mais eficiente foi a recursiva.

2.O programa abaixo implementa duas funções para multiplicar dois números, sendo uma interativa e outra recursiva. Apresente o teste de mesa em cada passo dos algoritmos para as seguintes chamadas:

```
I. mult(5,6)
II. multRec(5,6)

long int mult (int x, int y){
      long int res=0;
      while( y != 0){
            res += x;
            y--;
      }
      return(res);
}
long int multRec(int x, int y){
      if (y == 0){
            return 0;
      }else{
            return(x + multRec(x, y-1));
      }
}
```

R:

Na função comum

x = 5	y = 6	res = 0
res = 5	y = 5	x = 5
res = 10	y = 4	x = 5
res = 15	y = 3	x = 5
res = 20	y = 2	x = 5
res = 25	y = 1	x = 5
res = 30	y = 0	x = 5
res = 30		

Na função recursiva

x = 5	y = 6	
x = 5	y = 5	
x = 5	y = 4	
x = 5	y = 3	
x = 5	y = 2	
x = 5	y = 1	
x = 5	y = 0	Retorna 0

x = 5	y = 1
x = 5	y = 2
x = 5	y = 3
x = 5	y = 4
x = 5	y = 5
x = 5	y = 6
retorna 30

3.Faça um procedimento que incremente o valor de um número passado por parâmetro. Utilize a passagem de parâmetro por referência para fazer o incremento.

R:

```
void paraRefe(int *x){
    ++*x;
}
int main(){
    int x = 5;

    paraRefe(&x);

    printf("Numero com incremento: %d", x);
}
```

4.Faça um procedimento que receba 3 números inteiros A, B e C, e ordena os valores deforma que A passe a ter o menor valor e C o maior valor.

R:

```
int trocaValor(int a, int b, int c){
    int temp;
    if(a < b && a < c){
        temp = a;
    }else if(b < a && b < c){
        temp = b;
    }else{
        temp = c;
    }
    a = temp;

    if(a > b && a > c){
        temp = a;
    }else if(b > a && b > c){
        temp = b;
    }else{
        temp = c;
    }
    c = temp;
```

```
    printf("| A: %d | B: %d | C: %d |", a, b, c);
}
int main(){
    int a = 44, b = 188, c = 22;

    int res = trocaValor(a, b, c);
}
```

5.O que será impresso?

```
#include <stdio.h>
int main(int argc, char *argv[]) {
     int a=3, b=2, *p = NULL, *q = NULL;

     p = &a;
     q = p;
     *q = *q +1;
     q = &b;
     b = b + 1;
     printf("%d\n", *q);
     printf("%d\n", *p);
}
```

R: Será impresso 3 e 4.

6.Os programas (trechos de código) abaixo possuem erros. Qual(is)? Reescreva os códigos com as devidas correções.

```
(a)
int main(int argc, char *argv[]) {
     int x, *p;
     x = 100;
     p = x;
     printf("Valor dep: %d.\n", *p);
}
```

R: Como p é um ponteiro precisa receber um endereço de memória, no código com o erro p está recebendo o valor de x que causava erro.

```
int main(int argc, char *argv[]) {
     int x, *p;
     x = 100;
     p = &x;
     printf("Valor dep: %d.\n", *p);
}

(b)
void troca (int *i, int *j) {
     int *temp;
     *temp = *i;
     *i = *j;
     *j = *temp;
}
```

R: A variável temp foi declarada como um ponteiro que não está apontando para nenhum endereço de memória, então declarar temp como uma variável resolvera

o erro.

```
void troca (int *i, int *j) {
    int temp;

    temp = *i;
    *i = *j;
    *j = temp;
}
```

7.Qual o valor da variável x após a execução destas operações:

a)

```
int x = 2;
int *y = &x;
*y = 3;
printf("%d\n",x );
```

R: 3

b)

```
int x = 10;
int *y = &x;
int *z = &x;
int c = *y + *z;
*y = c;
printf("%d\n",x );
```

R: 20

c)

```
int x = 1;
x++;
int *y = &x;
*y = *y + 1;
printf("%d\n",x );
```

R: 3

d)

```
int x = 1;
x++;
int *y = &x;
y = y + 1;
printf("%d\n",x );
```

R: 2

8.Crie um procedimento que receba dois parâmetros: um vetor e um valor do mesmo tipo. O procedimento deverá preencher os elementos do vetor com esse valor. Não utilize índices para percorrer o vetor, apenas aritmética de ponteiros.

R:

```
void funcaoVetor(int num, int *vetor){
    if(num < 0){
        return 0;
    }else{
        funcaoVetor(num - 1, vetor);
        *(vetor+num) = num;
    }

    printf("--> %d\n", *(vetor+num));
}

int main(int argc, char *argv[]) {
    int vetor[20];
    int num = 20;

     funcaoVetor(num, &vetor);
}
```

9.Crie uma função que recebe um ponteiro para a posição inicial de uma matriz e recebe o tamanho da matriz. Preencha essa matriz com o valor 0 utilizando aritmética de ponteiros.

R:

```
int ponteiro(int *m, int tam){
    for(int i = 0; i < tam; i++){
        *(m+i) = 0;
    }
}
int main(int argc, char *argv[]) {
    int lin = 10, col = 10;
    int matriz[lin][col];
    int *p = matriz;
    int tam = lin * col;

    ponteiro(p, tam);

    for(int i = 0; i < lin; i++){
        for (int j = 0; j < col; j++){
            printf(" %d ", *(p+(i*col+j)) );
        }
        printf("\n");
     }
}
```

10.Crie uma função que imprime os valores das posições pares de um vetor utilizando aritmética de ponteiros para percorrer o vetor.

R: Estou considerando que o zero não seja par.

```
int funcaoVetor(int num, int *vetor){
    int mod;

    if(num < 0){
        return 0;
    }else{
        funcaoVetor(num - 1, vetor);
        *(vetor+num) = num;
    }

    mod = *(vetor+num) % 2;

    if(mod == 0 && *(vetor+num) > 0){
        printf("--> %d\n", *(vetor+num));
    }
}

int main(int argc, char *argv[]) {
    int vetor[20];
    int num = 20;

     funcaoVetor(num, &vetor);
}
```