Resolução de problemas de natureza discreta.

Atividade Avaliativa Recursão

Nome: Daniel Pereira Lima
Nome: Daniela Tamy Yuki
Nome: Eduardo Augusto Camacho
Nome: Isabella Lucena Conceição

Nome do Grupo: RA02 - Atividade Demonstração 2 **RA02 - Atividade Demonstração**

Entregar um PDF com os enunciados e as devidas soluções(link do git e replit) para cada questão.
A Interpretação das questões é parte da avaliação.
Coloque os conjuntos de teste nas soluções.
Início da atividade 08:10 - Fim:11:00 – Primeiros a sair a partir das 09:30.
As questões só serão validas se estiverem corretas por completo.

Aviso: os codigos não estão rodando no replit

1. Escreva um algoritmo recursivo para cada uma das alternativas (1,75).

a) $\begin{cases} S(1) = 10 \\ S(n) = S(n-1) + 10, \text{ para } n \geq 2 \end{cases}$

Em python:

```
def recursivo(n):
    if n == 1:
        return 10
    else:
        return recursivo(n-1)+10


n = int(input("Entre com um numero n:"))
resultado = recursivo(n)
print(resultado)
```

Em C:

```
#include <stdio.h>

int recursivo(int n){
    int resultado;
    if(n == 1){
        resultado = 10;
    }
    else{
        resultado = recursivo(n - 1) + 10;
    }

    return resultado;
}

int main(){
    int n, resultado;
    printf("Entre com o número N: "); scanf("%d", &n);
```

```
    resultado = recursivo(n);

    printf("%d\n", resultado);

    return 0;
}
```

b) $\begin{cases} A(1) = 2 \\ A(n) = A(n-1)^{-1}, \text{ para } n \geq 2 \end{cases}$

```
#include <stdio.h>

int recursivo(int n){
    int resultado;

    if(n == 1){
        resultado = 2;
    }
    else{
        resultado = recursivo(n - 1) * - 1;
    }

    return resultado;
}

int main(){
    int n, resultado;

    printf("Entre com o número N: "); scanf("%d", &n);

    resultado = recursivo(n);
    printf("%d\n", resultado);

    return 0;
}
```

c) $\begin{cases} B(1) = 1 \\ B(n) = B(n-1) + n^2, \text{ para } n \geq 2 \end{cases}$

```
#include <stdio.h>
#include <math.h>

int recursivo(int n){
    int resultado;
    if(n == 1){
        resultado = 1;
    }
    else{
        resultado = recursivo(n - 1) + pow(n,2);
    }

    return resultado;
}
```

```
int main(){
   int n, resultado;
   printf("Entre com o número N: "); scanf("%d", &n);

   resultado = recursivo(n);

   printf("%d\n", resultado);

   return 0;
}
```

d) $\begin{cases} P(1) = 1 \\ P(n) = n^{2*}P(n-1) + n - 1, \text{ para } n \geq 2 \end{cases}$

```
#include <stdio.h>

int recursivo(int n){
   int resultado;

   if(n == 1){
      resultado = 1;
   }
   else{
      resultado = (n*n)*recursivo(n - 1) + n - 1;
   }

   return resultado;

}

int main(){
   int n, resultado;

   printf("Entre com o número N: "); scanf("%d", &n);
   resultado = recursivo(n);

   printf("%d\n", resultado);

   return 0;
```

e) $\begin{cases} D(1) = 3 \\ D(2) = 5 \\ D(n) = (n-1)*D(n-1) + (n-2)*D(n-2), \text{ para } n > 2 \end{cases}$

```
#include <stdio.h>

int recursivo(int n) {
  int resultado;

  if (n == 1) {
    resultado = 3;
  } else if (n == 2) {
    resultado = 5;
  } else {
```

```
    resultado = (n-1) * recursivo(n - 1) + (n - 2) * recursivo(n - 2);
  }

  return resultado;
}

int main() {
  int n, resultado;

  printf("Entre com o número N: ");
  scanf("%d", &n);

  resultado = recursivo(n);
  printf("%d\n", resultado);

  return 0;
}
```

f) $\begin{cases} W(1) = 2 \\ W(2) = 5 \\ W(n) = W(n-1)*W(n-2), \text{ para } n > 2 \end{cases}$

```
#include <stdio.h>

int recursivo(int n) {
  int resultado;

  if (n == 1) {
    resultado = 2;
  } else if (n == 2) {
    resultado = 5;
  } else {
    resultado = recursivo(n - 1) * recursivo(n - 2);
  }

  return resultado;
}

int main() {
  int n, resultado;

  printf("Entre com o número N: ");
  scanf("%d", &n);

  resultado = recursivo(n);
  printf("%d\n", resultado);

  return 0;
}
```

g) $\begin{cases} T(1) = 1 \\ T(2) = 2 \\ T(3) = 3 \\ T(n) = T(n-1) + 2*T(n-2) + 3*T(n-3), \text{ para } n > 3 \end{cases}$

```
#include <stdio.h>
```

```
int recursivo(int n){
   int resultado;

   if(n == 1){
      resultado = 1;
   }
   else if(n == 2){
      resultado = 2;
   }
   else if(n == 3){
      resultado = 3;
   }
   else{
      resultado = recursivo(n - 1) + 2 * recursivo(n - 2) + 3 * recursivo(n - 3);
   }

   return resultado;
}

int main(){
   int n, resultado;
   printf("Entre com o termo N: "); scanf("%d", &n);

   resultado = recursivo(n);

   printf("%d\n", resultado);

   return 0;
}
```

2. Escreva uma definição recursiva para uma progressão geométrica com termo inicial a e razão r. (0,25)

```
#include <stdio.h>
#include <math.h>

int recursivo(int a, int q, int n) {
   int resultado;
   if (n == 1) {
      resultado = a;
   } else {
      resultado = a * recursivo(a, q, n - 1);
   }
   return resultado;
}

int main() {
   int a, q, n, resultado;
   printf("Entre com o número n: ");
   scanf("%d", &n);
   printf("Entre com o número a (termo inicial): ");
   scanf("%d", &a);
```

```
    printf("Entre com o número q (razão): ");
    scanf("%d", &q);

    resultado = recursivo(a, q, n);
    printf("%d\n", resultado);

    return 0;
}
```

3. Uma coleção T de números é definida recursivamente por:

$$\begin{cases} 2 \in T \\ \text{Se } X \in T, \text{ então} \quad X+3 \in T \quad \text{e} \quad 2*X \in T \end{cases}$$

Quais dos seguintes números pertencem a T? 6 , 7 , 19 , 12.

```
#include <stdio.h>

int pertence(int n) {

    if (n == 2) {

        return 1;

    }

    if (n > 2) {

        return pertence(n - 3) || pertence(n / 2);

    }

    return 0;

}

int main() {

    int numeros[] = {6, 7, 19, 12};

    int tamanho = sizeof(numeros) / sizeof(numeros[0]);

    printf("Tamanho: %d\n", tamanho);

    for (int i = 0; i < tamanho; ++i) {

        if (pertence(numeros[i])) {

            printf("%d pertence a T\n", numeros[i]);

        }

    }
```

```
    return 0;
}
```

4. Uma coleção M de números é definida recursivamente por:

$$\begin{cases} 2 \in M \text{ e } 3 \in M \\ \text{Se } X \in M \quad \text{e} \quad Y \in M, \text{ então} \quad X*Y \in M . \end{cases}$$

Quais dos seguintes números pertencem a M? 6 , 9 , 16 , 21 , 26 , 54 , 72 , 218.

Faça um programa recursivo para demonstrar. (0,25)

```
#include <stdio.h>

int pertence(int n) {
    if (n == 2 || n == 3) {
        return 1;
    }

    for (int i = 2; i <= n / 2; ++i) {
        if (n % i == 0 && pertence(i) && pertence(n / i)) {
            return 1;
        }
    }

    return 0;
}

int main() {
    int numeros[] = {6, 9, 16, 21, 26, 54, 72, 218};
    int tamanho = sizeof(numeros) / sizeof(numeros[0]);

    for (int i = 0; i < tamanho; ++i) {
        if (pertence(numeros[i])) {
            printf("%d pertence a M\n", numeros[i]);
        } else {
            printf("%d nao pertence a M\n", numeros[i]);
        }
    }

    return 0;
```

}

5. Uma coleção S de cadeias de caracteres é definida recursivamente por:

$$\begin{cases} a \in S \text{ e } b \in S \\ \text{Se } X \in S, \text{ então } Xb \in S. \end{cases}$$

Quais das seguintes cadeias pertencem a S? *a* , *ab* , *aba* , *aaab* , *bbbbb*

Faça um programa recursivo para demonstrar. (0,25)

6. Uma coleção W de cadeias de símbolos é definida recursivamente por:

$$\begin{cases} a \in W, \quad b \in W \text{ e } c \in W \\ \text{Se } X \in W, \text{ então } a(X)c \in W. \end{cases}$$

Quais das seguintes cadeias pertencem a S? *a(b)c* , *a(a(b)c)c* , *a(abc)c* , *a(a(a(a)c)c)c* , *a(aacc)c*

Faça um programa recursivo para demonstrar. (0,25)

7. Forneça uma definição recursiva para todas as cadeias binárias (cadeias formadas com os caracteres 0 e 1) contendo um número ímpar de zeros. (0,5 na prova)

8. Escreva o corpo da função recursiva para computar S(*n*) para uma dada seqüência S(1 ponto):

a) 1 , 3 , 9 , 27 , ...

b) 2 , 1 , $\frac{1}{2}$, $\frac{1}{4}$, ...

c) *a* , *b* , *a* + *b* , *a* + 2*b* , 2*a* + 3*b* , ...

d) *p* , *p* – *q* , *p* + *q* , *p* – 2*q* , *p* + 2*q* , *p* – 3*q* , ...

```
//a


#include <stdio.h>


int sequencia_a (int n){
   if (n == 1)
      return 1;
   else
      return (sequencia_a(n-1)*3);


}
int main(){
   int n;
   printf("insira o n-esimo numero: ");
   scanf("%d", &n);
   printf("[%d] O numero vai ser: %d", n, sequencia_a(n));
}
```

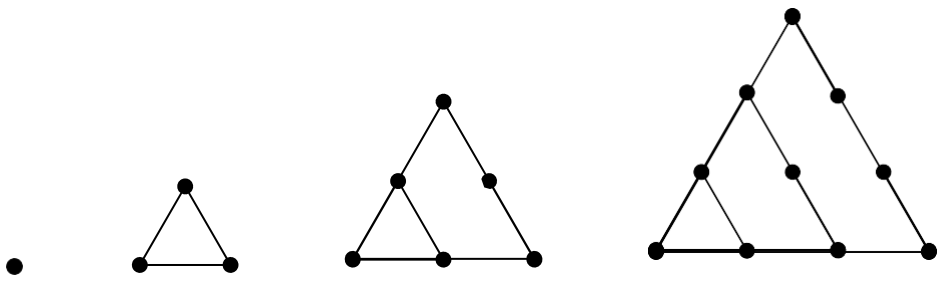
```
//b

#include <stdio.h>

double sequencia_b (int n){
    if (n == 1)
        return 2;
    else
        return (sequencia_b(n-1)/2);

}
double main(){
    int n;
    printf("insira o n-esimo numero: ");
    scanf("%d", &n);
    printf("[%d] O numero vai ser: %f", n, sequencia_b(n));
}
```

9. Membros antigos da Sociedade de Pitágoras definiram **números figurados** como sendo o número de pontos em uma certa configuração geométrica. Os primeiros **números triangulares** são 1, 3, 6 e 10, e são semelhantes ao diagrama da figura abaixo:



1 3 6 10

Encontre a fórmula para o *n*-ésimo número triangular e escreva um programa recursivo.(0,5)

```
#include <stdio.h>

int triangulo (int n){
   if (n == 1)
      return 1;
   else
      return (triangulo(n-1)+n);

}
int main(){
   int n;
   printf("insira o n-esimo numero: ");
   scanf("%d", &n);
   printf("[%d] O numero vai ser: %d", n, triangulo(n));
}
```

10 . Em um experimento, certa colônia de bactérias tem inicialmente uma população igual a 50.000. Uma leitura é feita a cada hora e, no final deste intervalo, há três vezes mais bactérias que antes. (a) Escreva a definição recursiva para A(*n*), o número de bactérias presentes no início do *n*-ésimo período de tempo. (0,25)(b) Em quantas leituras a população excederá 200.000 bactérias?(0,25)

```
#include <stdio.h>

int colonia (int n){
   if (n == 1)
      return 50000;
   else
      return (colonia(n-1)*3);

}
int main(){
   int n;
   printf("insira o n-esimo numero: ");
   scanf("%d", &n);
   printf("Em [%d] horas o numero vai ser: %d", n, colonia(n));
    }
```

11. (1,0) Considere o algoritmo recursivo:

```
Lista Rotina (Lista L, inteiro j) {
    Se (j == 1)
        return L;
    Encontre o L[i], o maior item da lista L entre 1 e j;
    Troque o L[i] pelo item L[j];
    return Rotina (L, j – 1);
}
```

Para L = [3, 7, 4, 2, 6] faça a chamada Rotina (L, 5);:

a) Represente L e o total de chamadas realizadas à Rotina