Pontifícia Universidade Católica de Minas Gerais

ICEI – Instituto de Ciências Exatas e Informática
DCC – Departamento de Ciência da Computação
Campus Lourdes
Bacharelado em Ciência da Computação

MAIOR UNIVERSIDADE CATÓLICA DO MUNDO - Fonte: Vaticano
MELHOR UNIVERSIDADE PRIVADA DO BRASIL - Guia do Estudante, por 6x
ENTRE AS MELHORES UNIVERSIDADES DO MUNDO - Times (Ranking Times High Education)
ÁREA DA COMPUTAÇÃO PUC MINAS: SEMPRE 1º..4º LUGAR PREF.MERCADO-Folha de S.Paulo (RUF), desde 2012
CIÊNCIA DA COMPUTAÇÃO PUC MINAS: SEMPRE 4 OU 5 ESTRELAS - Guia do Estudante
CIÊNCIA DA COMPUTAÇÃO *CAMPUS* LOURDES: NOTA MÁXIMA MEC - Av.Reconhecimento, 2023

Algoritmos e Estruturas de Dados I
Professor: Lúcio Mauro Pereira
Prova I - 23 de setembro de 2024
Valor: 25 pontos

Aluno(a): Adriano Araújo Henriques dos Santos **Nota:** 25

Antes de iniciar a avaliação, leia atentamente as instruções a seguir.

Desligue o seu **celular e guarde-o**.
Guarde também o seu relógio – a hora estará projetada no quadro.
A prova não poderá ser desgrampeada.
Prova **individual** e **sem consulta**.
Deixe sobre a carteira **apenas** caneta, lápis e borracha.
Não é permitido material algum sobre o **colo**, como blusas ou mochilas - guarde-os **debaixo da carteira**.
Não é autorizado o empréstimo de materiais durante a prova.

Caso alguma das regras acima seja violada, a prova será anulada.

Antes de desenvolver cada questão analise o problema, planeje uma solução, elabore um modelo de solução através de fluxogramas ou algoritmos. Em seguida, **codifique-o em Java**.

A correção incidirá apenas sobre a solução codificada.

A correção de cada questão considerará:

- o atendimento ao problema proposto;
- a qualidade da solução lógica;
- a codificação do programa;
- a endentação do código;
- a escolha adequada da estrutura de repetição;
- a documentação do programa.

Preocupe-se em consistir (validar) um dado lido apenas quando explicitamente solicitado pelo enunciado.

Não se preocupe em explicitar bibliotecas ou a instrução de encerramento do programa *(return 0)*.

Não haverá atendimento individual durante a prova. A interpretação das questões faz parte da avaliação. Fique à vontade para justificar, junto à questão, as decisões que tomar durante a solução.

Tempo estimado para realização da prova: 65 minutos.

5

(5,0) Objeto de avaliação: **estruturas de seleção e repetição, modularização, parâmetros**
Construa uma função que verifique se dois números inteiros são *primos entre si*.
Dois números inteiros são ditos *primos entre si* caso não exista divisores comuns a ambos, exceto o número 1.
Observe que para dois números serem *primos entre si* não é necessário serem eles *números primos*. Ao contrário, a única coisa a ser observada é comportamento de ambos quanto aos seus divisores.
Parâmetros: dois inteiros, relativos aos dois números a serem comparados.
Valor gerado: *verdadeiro*, se os dois números naturais forem primos entre si, ou *falso*, caso contrário.
Obs: se um dos inteiros for o valor *zero*, a função deverá gerar *falso*.

```
bool PRIME(INT A, INT B){
    bool isPRIME = TRUE;
    IF(A==0 || B==0){
        isPRIME = FALSE;
    }ELSE{
        INT SMALLEST = A<B? A:B;
        INT FLAG = SMALLEST/2;  // DEVERIA SER 'SQRT(SMALL)'
                                   POR QUESTÃO DE OPTIMIZAÇÃO
        INT i = 2;
        WHILE(i<=FLAG && isPRIME){
            bool ADIVISIBLE = A%i==0;
            bool BDIVISIBLE = B%i==0;
            IF(ADIVISIBLE && BDIVISIBLE) isPRIME = FALSE;
            i++;
        }
    }
    RETURN isPRIME;
}
```

melhor entes

Objeto de avaliação: **estruturas de seleção e repetição, modularização, parâmetros, abordagem iterativa *versus* abordagem recursiva**

Construa uma função que calcule o produto entre dois números (multiplicação) utilizando unicamente a operação de adição – considere proibido aqui o uso do operador *, utilizando apenas o perador +. Planeje, cuidadosamente, os parâmetros a serem encaminhados e o valor gerado pela função.

a) *(5,0)* Abordagem iterativa

b) *(5,0)* Abordagem recursiva

```
INT PRODUCT (INT A, INT B){ // ITERATIVA
    INT RESULT = 0;
    FOR (INT i = 0; i < B; i++){
        RESULT += A;
    }
    RETURN RESULT;
}
```

```
INT PRODUCT (INT A, INT B){// RECURSIVA
    INT RESULT = 0;
    IF (B > 0){
        RESULT = A + PRODUCT(A, B-1);
    }
    RETURN RESULT;
}
```

Objeto de avaliação: **estruturas de seleção e repetição, modularização, parâmetros, abordagem iterativa *versus* abordagem recursiva**

A série de FETUCCINE é gerada da seguinte forma: os dois primeiros termos são dados; a partir daí, os termos seguintes são gerados com a soma ou subtração dos dois termos anteriores, seguindo o comportamento abaixo:

$A_i = A_{i-1} + A_{i-2}$, *para i par*

$A_i = A_{i-2} - A_{i-1}$, *para i ímpar*

Em outras palavras, quando em posição par, um novo número é obtido pela soma dos dois anteriores; por outro lado, quando em posição ímpar, um novo número é obtido subtraindo o número anterior do penúltimo.

Exibir na tela do monitor de vídeo os *n* primeiros termos da série.

Planeje, cuidadosamente, os parâmetros a serem encaminhados e o valor gerado pela função. De imediato, note que os dois primeiros termos, bem como o número de termos, precisam ser parametrizados.

a) *(5,0)* Abordagem iterativa

b) *(5,0)* Abordagem recursiva

CÓDIGO NO VERSO

```
VOID FETUCCINE(INT TERM1, INT TERM2, INT N){

IF(N >= 1) PRINTF("\n%d", TERM1);
IF(N >= 2) PRINTF("\n%d", TERM2);
IF(N >= 3) {3
    FOR(INT i = 3 ; i <= N ; i++){
        bool isEVEN = i % 2 == 0;
        INT TERM;
        IF(isEVEN){
            TERM = TERM2 + TERM1;
        } ELSE {
            TERM = TERM1 - TERM2;
        }
        PRINTF("\n%d", TERM);
        TERM1 = TERM2;
        TERM2 = TERM;
    }
}
```

```
int GetTerm (int prev1, int prev2, int N){
    int Term;
    bool isEven = N % 2 == 0;

    if (isEven) {
        Term = prev1 + prev2;
    } else {
        Term = prev2 - prev1;
    }
    return Term;
}

void Fetuccine (int Term1, int Term2, int N){
    if (N == 1) {
        printf("\n%d", Term1);
    } else if (N == 2) {
        printf("\n%d", Term2);
    } else {
        int Term = GetTerm(
            GetTerm(Term2, Term1, N-1),
            GetTerm(Term2, Term1, N-2),
            N
        );

        printf("\n%d", Term);
}
```