Fatec São Caetano do Sul Antonio Russo

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

Análise e Desenvolvimento de Sistemas

Manhã – ADSMA3

Estrutura de Dados

Atividade sobre números aleatórios e recursividade

Felix Petiz Bonilho RA:168048191002

Matheus Barbosa do Nascimento RA:1680481911008

Thainá Araújo Mendonça RA:1680481811044

Victor Hugo Trindade Tenedini RA:1680481911022

05/2020

Atividade sobre Números Aleatórios e Recursividade

Realizar os exercícios em grupos de até 4 pessoas. Não serão aceitas atividades individuais!!

Apresentar **apenas** as funções solicitadas nos exercícios (as funções devem ser comentadas) em um arquivo PDF contendo o nome completo dos participantes. Embora todas recebam, apenas 1 elemento do grupo deve postar a tarefa.

O prazo de entrega desta atividade (tarefa) é 31/05 até às 23:59.

1. Preencher um vetor de tamanho 10 com valores também variáveis na faixa de 1 a 10, mas sem repetição (tam=10)

   Protótipo: **void semRepetidos(int v[], int tam)**

```c
void semRepetidos (int v[], int tam){
    srand(time(NULL)); // inicializando o procedimento randômico
    int i;
    int j;
    tam=rand()%10+1 // tamanho entre 1 e 10
    int v[tam];
    for(i=0;i<tam;i++) {  //preenche o vetor com 10 posições
        v[i]= rand()% 10+1; //faz o sorteio randomico entre 1 e 10
        for(j=0; j<i; j++) { //verifica se o número já existe
            if(vetor[j] == vetor[i]) {
                vetor[i] = (rand()%i)+1;
                j=0;
            }
        }
        printf("%d\t",v[i]);
    }
    printf("\n\n");
    system("pause");
}
```

2. Escreva uma **função recursiva** que faça a procura sequencial de um valor passado por parâmetro em um vetor (preenchido sem repetição) também passado por parâmetro. A função deve retornar o índice se encontrado, ou -1 se não encontrado.
   Protótipo: **int buscavetor (int *vet, int tam, int valor)**

```c
void buscaNota(Lista *ls, int mat) { //recebe a lista (ls) e a matrícula a ser procurada (mat)
    if (ls == NULL) //Verifica se o elemento atual é o último da lista
        printf("\nMatricula Inexistente\n\n"); //imprime que não encontrou a matrícula
    else if (ls->matricula == mat) //verifica se a matrícula atual é igual à informada
        printf("\nNota referente a matricula %d = %.2f\n\n", mat, ls-
>nota); //imprime a nota refrênte a matrícula atual
    else //caso a matrìcula atual não seja igual a informado e este não for o último elemento
```

```
        buscaNota(ls->next, mat); //chama a função passando o proximo elemento da lista
}
```

Considere: typedef struct lista{
int matricula;
float nota;
struct lista *prev;
struct lista *next;
}Lista;

3. Escreva uma **função recursiva** que procure uma matrícula(**mat**) passada por parâmetro em uma lista duplamente encadeada (**ls**) cujo ponteiro para o valor inicial foi passado por parâmetro. A função deve mostrar a **nota** referente à matrícula informada ou informar matrícula inexistente.
   Considere: typedef struct lista{
   int matricula;
   float nota;
   struct lista *prev;
   struct lista *next;
   }Lista;

   Protótipo: **void buscaNota (Lista *ls, int mat)**

```
void buscaNota(Lista *ls, int mat) { //recebe a lista (ls) e a matrícula a ser procurada (mat)
    if (ls == NULL) //Verifica se o elemento atual é o último da lista
        printf("\nMatricula Inexistente\n\n"); //imprime que não encontrou a matrícula
    else if (ls->matricula == mat) //verifica se a matrícula atual é igual à informada
        printf("\nNota referente a matricula %d = %.2f\n\n", mat, ls-
>nota); //imprime a nota refrênte a matrícula atual
    else //caso a matrìcula atual não seja igual a informado e este não for o último elemento
        buscaNota(ls->next, mat); //chama a função passando o proximo elemento da lista
}
```

4. Usando a mesma lista acima, escreva uma **função recursiva** que receba o último elemento da lista (**ult**) e uma **nota** e retorne o número de matrículas associadas a nota passada por parâmetro.

   Protótipo: **int contaNotas (Lista *ult, float nota) {**

```
int contaNotas(Lista *ult, float nota){
    int cont = 0;  // declara a váriavel para contar a quantidade de notas
    if(ult != NULL){ // verifica se existe a lista
        if(ult->nota == nota){ // verifica se a nota da lista é igual a nota passada
            cont++; // se for igual, soma +1 no cont
        }
        if(ult->prev != NULL){ // verifica se existe uma lista anterior
```

```
            cont += contaNotas(ult-
>prev, nota); // se existir, chama a função recursivamente e soma o resultado em cont
        }
        return cont; // retorna o cont
    } else return 0; // retorna 0 caso não exista uma lista
}
```