Curso de Ciência da Computação - Estrutura de Dados I – 2024.1
OBS: Obrigatório utilizar o protótipo de função apresentado e os tipos de dados utilizados nas aulas

1. Faça um algoritmo que recebe duas matrizes nxn armazenadas nos vetores v1 e v2 e, calcula uma nova matriz a ser armazenada no vetor v3 que corresponde a multiplicação da transposta da matriz armazenada no vetor v1 pela matriz armazenada no vetor v2. Considere que o vetor v3 já está devidamente alocado. Não pode usar matrizes ou vetores auxiliares.
Obs: não é para tentar transpor a matriz armazenada no vetor v1.
int MultiplicaMatrizes (int *v1, int *v2, int *v3, int n)

2. Faça um algoritmo que recebe uma fila implementada como um vetor circular e remove todos os elementos de ordem par (segundo, quarto, sexto, etc.) Não pode utilizar um vetor auxiliar ou outra estrutura de dados.
TAD
int RemoveElementosOrdemParFilaCircular (Queue *q, int n) ignorar o n

3. Faça um algoritmo que recebe uma pilha armazenada em um vetor, um valor (chave), e uma função de comparação, e remove respeitando a disciplina de acesso da pilha todos os elementos até encontrar um com chave menor que o valor da chave recebida. Não pode usar pops e push, e deve obedecer a disciplina de acesso da pilha. É uma função interna do TAD Pilha.
TAD
int RemoveMaioresQueKey (Stack *s, void *key, int (*cmp) (void *, void *))
OBS: cmp (a,b) retorna TRUE se a <b e False caso contrário

4. Faça um algoritmo que recebe dois vetores de caracteres s1 e s2 com uma frase em que as palavras estão separadas por espaços em branco e, usando uma única pilha, verificar se as palavras que existem em s1 estão em s2 na mesma sequencia mas em ordem inversa. Não pode usar memória auxiliar somente usar as funções do TAD (stkCreate, stkPop, stkPush, stkDestroy).
MAIN
int VerificaString(Stack *s1, char *s2, int length);
Exemplo de strings que tornam verdadeira a afirmativa:
S1 = "O Flamengo é o melhor time do Brasil"
S2 = "O ognemalF é o rohlem emit od lisarB"

06/05/24

90/100

1.
```c
int multiplicamatrizes(int *V1, int *V2, int *V3, int n){
    if(V1!=null && V2!=null && V3!=null && n>0){
        for(int i=0; i<n; i++){
            for(int j=0; j<n; j++){
                V3[i*n+j]=0;
                for(int k=0; k<n; k++){
                    V3[i*n+j] += V1[k*n+i]*V2[k*n+j];
                }
                return true;
            }
        }
    }
    return false;
}
```

30/30

// a transposta de V1[i*n+j] é V1[j*n+i]; enquanto V1 na multiplicação é V1[i*n+k] logo, a transposta na multiplicação é V1[k*n+i] \\

2.
```c
int removeelementosordemporfilacircular(queue *q){
    if(q!=null){
        int atual = q->rear;
        for(int i=1; i<q->~~maxItens~~ rear; i+=2){
            prox = inccirc(i, q->maxItens);
            q->elems[i] = q->elems[prox];
            q->nelems--;
            atual--;
        }
        q->rear = atual;
        return true;
    }
    return false;
}
```

00/30

// a função identifica os elementos de ordem par, e define eles como sendo seu sucessor, enquanto decrementa o número de elementos, e o rear, em uma posição auxiliar. //

```c
int inccirc(int atual, int max){
    if(atual < max){
        atual++;
        return atual;
    }else{
        atual=0;
        return atual;
    }
```

06/05/24

3. int RemoveAllKey(Stack *s, void *key, int (*cmp)(void*, void*)){
    if(s!=null && key!=null && cmp!=null){
        int i = s->top;
        while(cmp(key, s->item[i])!=false){
            s->top--;
            i--;
            if(cmp(key, s->item[i])!=false && i==0){
                return false;
            }
        }
        return true;
    }
    return false;
}

// a função inicia em s->top, e
enquanto a chave é maior que o
item atual, implanta for, o topo é
decrementado, ocorrem comparações.
Se i é igual a zero e a chave
ainda é maior que o item atual,
a retorna é falso, caso saia do
loop, a retorna é verdadeira. //

int cmp(void *a, void *b){
    int *ptra = (int*) a;
    int *ptrb = (int*) b;
    if(*ptra > *ptrb){
        return true;
    }else{
        return false;
    }

#define true 1
#define false 0

#define true 1
#define false 0

06/05/24

```
4. int VerificaString(char* s1, char* s2, int length){
       if(s1!=null && s2!=null && length>0){
           Stack *pilha=(stack*)malloc(sizeof(Stack)*length);
           if(pilha!=null){
               int i=0;
               int j=0;
               int stat=true;
               char* caractere;
               While(stat!=false){
                   While(s1[i]!=' '){
                       stackpush(pilha,(void*)&str[i]);
                       i++;
                   }
                   i++;
                   While(s2[j]!=' '){
                       caractere=(char*)stackpop(pilha);
                       if(caractere!=s2[j]){
                           return false;
                       }else{
                           j++;
                       }
                   }

                   if(caractere==null && i==strlen(s1)-1 && j==strlen(s2)-1){
                       stat=false;
                   }
               }
               return true;
           }
       }
       return false;
   }
```

30/30

// a função verifica se os elementos de s1 são diferentes do espaço branco enquanto for, armazena na pilha. Enquanto o elemento tirado da pilha for igual a s2[j], j é incrementado até quebrar o laço. Se caractere é nulo, i é o maior índice de S1 e j é o maior índice de S2, o laço maior é quebrado e o retorno é verdadeiro.