

Prova 3
Programação e Desenvolvimento de Software I
Professor: Pedro O.S. Vaz de Melo

Nome: _____
escrevendo o meu nome eu juro que seguirei o código de honra

Código de Honra para este exame:

- Não darei ajuda a outros colegas durante os exames, nem lhes pedirei ajuda;
- não copiarei nem deixarei que um colega copie de mim;
- não usarei no exame elementos de consulta não autorizados.

Referências:

Função/Operador	Descrição	Exemplo
<code>void* malloc (size_t size);</code>	aloca um bloco de memória de tamanho <code>size</code> , retornando um ponteiro para o início do bloco.	<code>int *p1 = (int*)malloc(sizeof(int));</code>
<code>void free (void *p);</code>	Desaloca o bloco de memória apontado por <code>p</code> .	<code>free(p);</code>

1. (7 points) Um *triângulo cortado* de altura h e lateral i é um triângulo formado por asteriscos e com as suas laterais cortadas, de forma a fazer com que os seus lados tenham altura i . Abaixo um exemplo de um *triângulo cortado* de altura 5 e lateral 2:

```
  *
 ***
*****
*****
*****
```

Escreva uma função RECURSIVA para contar o número total de asteriscos usados para desenhar um *triângulo cortado* de altura h e lateral i . Para o exemplo dessa questão, a sua função deve retornar 23. Sua função não pode usar nem *loops* (`for`, `while`, `etc`) nem variáveis globais. Protótipo:

```
int numAsts(int i, int h);
```

2. (5 points) Escreva uma função RECURSIVA que recebe um ponteiro para uma *string* como parâmetro e retorna o seu tamanho. Sua função não pode usar nem *loops* (`for`, `while`, `etc`) nem variáveis globais. Protótipo:

```
int tamString(char *str);
```

3. (5 points) Escreva um procedimento que recebe um ponteiro para caractere como **parâmetro por referência** e aloca para esse ponteiro uma *string* de tamanho e conteúdo aleatório. O tamanho deve ser entre 1 e 10. As posições da *string* devem ser preenchidas com caracteres entre 'A' (65) e 'Z' (90). O seu procedimento deve ter o seguinte protótipo:

```
void fillStr(char **str);
```

4. (10 points) Complete o código abaixo, que descreve uma função de nome `merge` que junta duas *strings* `str1` e `str2` em uma única *string* e **retorna** o ponteiro para essa nova *string*. A sua função deve alocar o espaço da nova *string* (primeiros três espaços), transferir o conteúdo das *strings* passadas como parâmetro para esse novo espaço (cinco espaços seguintes) e liberar o espaço das duas que foram passadas como parâmetro. Dica: `n1` e `n2` são os tamanhos das strings apontadas por `str1` e `str2`, respectivamente.

```
char* merge(char *str1, char *str2) {

    int n1 = _____; //0.5

    int n2 = _____; //0.5

    char *str = _____; //1
    int i = 0;

    for(_____) //1

        str[_____] = str1[_____]; //1

    for(_____) //1

        str[_____] = str2[_____]; //1

    _____; //1

    _____; //1

    _____; //1

    _____; //1

}
```

5. (3 points) Complete o código abaixo, que descreve uma programa para testar a função `merge` do exercício anterior.

```
void main() {
    char *s1, *s2;

    fillStr(_____); //1

    fillStr(_____); //1

    char *s3 = merge(_____); //1
    printf("\n%s", s3);
}
```