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
`int abs(int x)`	retorna $\|x\|$	`abs(-3)` retorna $\|3\| = 3$
`void* malloc (size_t size);`	aloca um bloco de memória de tamanho `size`, retornando um ponteiro para o início do bloco.	`int *p1 = (int*)malloc(sizeof(int));`
`void free (void *p);`	Desaloca o bloco de memória apontado por `p`.	`free(p);`

1. (*5 points*) Suponha que a sua linguagem de programação suporte apenas operações de soma e subtração. Assim, escreva uma função RECURSIVA para calcular e retornar a multiplicação de n por k, os dois pertencentes ao conjunto dos números naturais ($\{0, 1, 2, \cdots\}$). Sua função não pode usar *loops* (`for, while, etc`), nenhuma função da biblioteca `math.h`, e nem os operadores de multiplicação e divisão. Protótipo:

```
int multi(int n, int k);
```

2. (*5 points*) Escreva uma função RECURSIVA que recebe um vetor de inteiros **ORDENADO** (em ordem crescente) e o seu tamanho n e retorna a menor diferença absoluta entre quaisquer dois pares desse vetor. A diferença absoluta entre dois números é o módulo da diferença entre esses dois números, de forma que a ordem em que a subtração é efetuada não importa. Sua função não pode usar *loops* (`for, while, etc`). Ex: se o vetor for {`-5, 1, 4, 9`} a sua função deve retornar `3`. O seu procedimento deve ter o seguinte protótipo:

```
int menorDif(int v[], int n);
```

3. (*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 *loops* (`for, while, etc`) e deve ter o seguinte protótipo:

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

4. (*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);
```

5. (*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
}
```

6. (*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);
}
```