Universidade Federal do Maranhão
Departamento de Informática
Ciência da Computação
Linguagem de programação
Prof.: Francisco Glaubos

Lista 2

strings, ponteiros, structs, arquivos, gerenciamento de memória e listas encadeadas

1 Strings

Questão 1.1:

Se as duas strings forem idênticas, a função strcmp () retorna

a) -1

b) 1

c) 0

d) yes

Questão 1.2:

```
#include <stdio.h>
#define MAX_SIZE 100

int func(const char * str, const char x);

int main(){
    char str[MAX_SIZE];
    char x;
    int y;
    gets(str);
    x = getchar();
    y = func(str, x);
    printf("%d", y);
    return 0; }


int func(const char * str, const char x){
    int y = -1;
    int i = 0;
    while(str[i] != '\0')     {
        if(str[i] == x)        {
```

```
            y = i;
        }
        i++;
    }
    return y; }
```

O que faz o programa em C escrito acima?

a) encontra a última ocorrência de um caractere em uma determinada string.

b) encontra a primeira ocorrência de um caractere em uma determinada string.

c) encontra todas as ocorrências de um caractere em uma determinada string.

d) conta as ocorrências de um caractere em uma determinada string.

Questão 1.3:

Escreva um programa C para converter uma string com letras minúsculas em uma string com letras maiúsculas.

Questão 1.4:

Escreva um programa em C para verificar se uma string é palíndromo ou não.

Questão 1.5:

Escreva um programa em C para inverter a ordem das palavras em uma determinada string.

Questão 1.6:

Escreva um programa em C para encontrar a primeira ocorrência de uma palavra em uma determinada string.

Questão 1.7:

Escreva um programa em C para remover todos os espaços em branco extras de string de entrada.

Questão 1.8:

Escreva um programa em C para remover todos os caracteres repetidos de uma determinada string.

Questão 1.9:

Escreva um programa em C para contar as ocorrências de uma palavra em uma determinada string.

Questão 1.10:

Utilizando ponteiros, escreva uma função em C para retornar o primeiro caractere não repetido em uma determinada string passada como argumento.

Exemplo de 3 entradas:	*Saídas:*
‘uihsad’	‘u’
‘uuqiqoq’	‘i’
‘ggghhh’	Nenhum

2 Ponteiros

Questão 2.1:

Escreva um programa em C para imprimir uma string ao contrário usando um ponteiro

Questão 2.2:

Considere um compilador onde int leva 4 bytes, char leva 1 byte e o ponteiro leva 4 bytes. Qual será a saída do seguinte programa em C?

```
#include <stdio.h>

int main()
{
    int arri[] = {1, 2 ,3};
    int *ptri = arri;

    char arrc[] = {1, 2 ,3};
    char *ptrc = arrc;

    printf("sizeof arri[] = %d ", sizeof(arri));
    printf("sizeof ptri = %d ", sizeof(ptri));

    printf("sizeof arrc[] = %d ", sizeof(arrc));
    printf("sizeof ptrc = %d ", sizeof(ptrc));

    return 0;
}
```

a) sizeof arri[] = 3 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 4

b) sizeof arri[] = 12 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 1

c) sizeof arri[] = 3 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 1

d) sizeof arri[] = 12 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 4

Questão 2.3:

Crie uma função em C para trocar os valores de dois números passados por referência.

Ex.:
Antes da chamada da função: a=1 e b=2
Após a chamada da função: b=1 e a=2

Questão 2.4:

Escreva um programa C para buscar um elemento em um vetor, usando ponteiros.

Questão 2.5:

Escreva um programa em C para imprimir todos as letras do alfabeto usando um ponteiro.

Questão 2.6:

Qual será a saída do seguinte programa em C?

```c
#include<stdio.h>
int main()
{
  char *ptr = "helloworld";
  printf(ptr + 4);
  return 0;
}
```

a) oworld b) world c) hell d) hello

Questão 2.7:

Qual será a saída do seguinte programa em C?

```c
#include<stdio.h>
int main()
{
  char *ptr = "auladelp";
  printf("%c\n",*&*ptr);
  return 0;
}
```

a) Endereço de 'a' b) Erro de compilação c) a d) Erro de execução

Questão 2.8:

Qual será a saída do seguinte programa em C?

```c
#include<stdio.h>
int main(){
  int a = 25, b;
  int *ptr, *ptr1;
  ptr = &a;
  ptr1 = &b;
  b = 36;
  printf("%d %d",*ptr, *ptr1);
  return 0;
}
```

a) 25 45632845

b) Erro de execução

c) Erro de compilação

d) 25 36

Questão 2.9:

Qual será a saída do seguinte programa em C?

```c
#include<stdio.h>
int main(){
  int a = 36;
  int *ptr;
  ptr = &a;
  printf("%u %u", *&ptr , &*ptr);
  return 0;
}
```

a) Endereço Valor

b) Valor Endereço

c) Endereço Endereço

d) Erro de compilação

Questão 2.10:

Qual será a saída do seguinte programa em C?

```c
#include<stdio.h>
int main(){
  int i = 25;
  int *j;
  int **k;
  j = &i;
  k = &j;
  printf("%u %u %u ",k,*k,**k);
  return 0;
}
```

a) endereço endereço valor

b) endereço valor valor

c) endereço endereço endereço

d) erro de compilação

Questão 2.11:

Qual será a saída do seguinte programa em C?

```c
int main() {
  int v[]={0};
  v[0]=1;
  free(v);
  printf("%d", v[0]);
return 0;
}
```

a) erro de compilação

b) erro de execução

c) 0

d) 1

Questão 2.12:

Qual o tipo de retorno da função *malloc()*?

a) void*

b) ponteiro do tipo da memória alocada

c) void**

d) int*

Questão 2.13:

Crie um programa que solicite ao usuário números até que um caractere zero seja inserido. Esses números serão armazenados em um vetor `*vetNum` (inicialmente de tamanho igual a zero). Cada vez que um novo valor é introduzido, o bloco de memória apontado por `*vetNum` é aumentado pelo tamanho de um *int*.

Questão 2.14:

Crie um programa em C para simular um carrinho de compras com tamanho flexível.

Utilizando alocação dinâmica de memória, crie funções para adicionar e remover nomes de produtos em uma variável char* carrinho. A capacidade do carrinho deve ser inicialmente igual a 0, e deve ser aumentada em 1 unidade antes de cada nova adição de produto.

3 Structs

Questão 3.1:

Qual operador conecta o nome de uma *struct* ao seu nome de membro?

a) -

b) <-

c) .

d) Ambos <- e .

Questão 3.2:

Qual dos seguintes não pode ser um membro de uma *struct*?

a) Outra estrutura

b) Função

c) Matriz

d) Nenhum dos mencionados

Questão 3.3:

Escreva um programa em C para manter registros e realizar análises estatísticas para uma turma de 20 alunos. As informações de cada aluno contêm ID, nome, sexo, pontuação dos testes (2 testes por semestre) e pontuação total
O programa solicitará que o usuário escolha a operação de registros em um menu, como mostrado abaixo:

```
================================================
                    Menu
================================================
1. Adicionar registros de estudante

2. Deletar registros de estudante

3. Atualizar registros de estudante

4. Ver registros de todos os estudantes

5. Mostrar aluno com maior nota total

6. Mostrar aluno com menor nota total

7. Encontrar aluno por ID

8. Ordenar os registros por pontuacao total

Entre com a sua opção:
```

Obs.: Todos os registros de aluno devem ser armazenados em vetores de *structs*

Questão 3.4:

Crie uma struct para representar uma fração, na qual um campo representa o numerador e outro o denominador da fração.
Em seguida crie uma função para somar duas frações e exibir a fração do resultado. Seu programa solicitará que o usuário insira a fração 1 e a fração 2. O numerador e o denominador de cada fração são inseridos separadamente pelo espaço. Veja o exemplo de saída abaixo:

```
Digite a fração 1 (denominador do numerador): 1 2
```

Digite a fração 2 (denominador do numerador): 2 5
Resultado: 9/10

Questão 3.5:

Defina um tipo de estrutura para representar um ponto através de suas coordenadas cartesianas. Em seguida, crie uma função para calcular e retornar a distância euclidiana entre dois pontos fornecidos como entrada.

Dica: a distância entre dois pontos P e Q, é dada pela seguinte fórmula:

$d_{qp} = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}$

Questão 3.6:

Crie uma struct para representar uma Pessoa, com nome (até 100 caracteres), endereço (até 200 caracteres) e telefone; em seguida, crie uma agenda para armazenar o contato de até n pessoas (o espaço da agenda deverá ser alocado dinamicamente).

Em seguida, crie uma função `void addPessoa(Pessoa* agenda, int n)` que adicione uma pessoa à agenda;

Questão 3.7:

A partir do exercício anterior, crie uma função para buscar uma pessoa na agenda (pelo nome) e retornar o número do seu telefone.

Questão 3.8:

```
typedef struct {
  char modelo[100];
  int ano;
}Carro;
```

Crie um programa em C para o cadastro de carros a partir do teclado em um vetor dinâmico (`Carro* catalogo`), inicialmente de tamanho igual a zero. Cada vez que um novo carro for introduzido, o bloco de memória apontado por `catalogo` é aumentado pelo tamanho de um `Carro`. O cadastro de um carro é realizado iterativamente em um loop até que um ano *negativo* seja dado como entrada no campo `Carro.ano`.

Dica: utilize as funções `void* malloc(unsigned size)` e `void* realloc(void* ptr, unsigned size)`

Questão 3.9:

Vamos trabalhar no menu de uma biblioteca. Crie uma *struct* contendo informações de livros como número de registro, nome do autor, título do livro e um campo `ocupado` para saber se o livro foi alugado ou não.

O biblioteca deve ser iniciada como vazia, e ir crescendo dinamicamente à medida que novos livros são cadastrados.

Crie um menu com funções para:

1. Exibir informações de um livro: `void exibirLivro(Livro* biblioteca, char* titulo)`
2. Adicionar um novo livro: `void addLivro(Livro* biblioteca, Livro novo)`
   (a) Dica: utilizar `void* realloc(void* ptr, unsigned size)`
3. Exibir todos os livros da biblioteca de um determinado autor: `void mostrarBiblioteca(Livro* biblioteca)`
4. Exibir o número total de livros na biblioteca

4 Arquivos

Questão 4.1:

Quando fopen() não é capaz de abrir um arquivo, ele retorna:

a) EOF
b) NULL
c) Erro de execução
d) Erro de compilação

Questão 4.2:

Escreva um programa em C para contar caracteres, palavras e linhas de um arquivo de texto.

Questão 4.3:

Escreva um programa em C para remover uma palavra de um arquivo de texto.

Questão 4.4:

Escreva um programa C para imprimir na tela o seu próprio código-fonte.

Questão 4.5:

Escreva um programa C para remover linhas vazias de um arquivo de texto.

Questão 4.6:

Escreva um programa em C para localizar e substituir uma palavra em um arquivo de texto.

Questão 4.7:

Escreva um programa C para renomear um arquivo usando a função *rename()*.

Questão 4.8:

Qual será a saída do seguinte programa em C?

```c
#include<stdio.h>
int main()
{
  int EOF = 0;
  printf("%d", EOF);
  return 0;
}
```

a) -1

b) 0

c) 1

d) Erro de compilação

Questão 4.9:

Escreva um programa em C para substituir uma linha específica por outro texto em um arquivo. O usuário deverá entrar com o nome do arquivo, o número da linha a ser substituída e a novo texto.

Ex. de entrada:

```
Arquivo:
teste 1
teste 2
teste 3

Texto a ser substituído: "teste 2"
Linha: 2
```

Saída:

```
Arquivo:
teste 1
teste 3
```

Questão 4.10:

Qual será a saída do seguinte programa em C? Paral tal, considere que o arquivo data.txt contém o seguinte conteúdo: *Aula de LP.*

```c
#include<stdio.h>
int main()
{
  unsigned char ch;
  FILE *fp;
  fp = fopen("data.txt", "r");
  while ((ch = fgetc(fp)) != EOF)
  {
    printf("%c", ch);
  }
  printf(" Obrigado.");
  fclose(fp);
  return 0;
}
```

a) erro de compilação

b) nada será mostrado

c) "Aula de LP. Obrigado."

d) "Aula de LP". E o loop continua infinitamente.

Questão 4.11:

Qual será a saída do seguinte programa em C? Paral tal, considere que o arquivo data.txt contém o seguinte conteúdo: *Aula de LP.*

```c
#include<stdio.h>
int main() {
  char ch;
  FILE *fp;
  fp = fopen("data.txt", "w");
  while ((ch = fgetc(fp)) != EOF)
  {
    printf("%c", ch);
  }
  printf(" Obrigado.");
  fclose(fp);
  return 0;
}
```

a) erro de compilação

b) "Obrigado"

c) "Aula de LP. Obrigado."

d) "Aula de LP". E o loop continua infinitamente.

Questão 4.12:

O código de César é uma das mais simples e conhecidas técnicas de criptografia. É um tipo de cifra de substituição na qual cada letra do texto é substituída por outra, que se apresenta no alfabeto abaixo dela um número fixo de vezes (k). Considera-se a lista de alfabeto como sendo circular. Por ex.: com $k = 3$, $'A'$ seria substituído por $'D'$, $'z'$ se tornaria $'c'$, e assim por diante.

a) Utilizando o código de César, crie um programa em C que leia senhas de um arquivo (senhas.txt) contendo senhas de até n caracteres, e as mostre criptografadas na tela.

b) Utilizando o código de César, crie um programa em C que leia senhas criptografadas de um banco de dados (bd.txt) contendo senhas de até n caracteres, e as mostre descriptografadas na tela.

Questão 4.13:

Escreva um programa em C para mesclar dois arquivos e escrever o resultado da mesclagem em um novo arquivo.

Arquivo 1:
aaaaaa
bbbbbb

Arquivo 2:
cccccc
dddddd

Entrada:
nome_arquivo1.txt nome_arquivo2.txt nome_arquivo3.txt

Saída:

Arquivo 3:
aaaaaa
cccccc
bbbbbb
dddddd

5 Desafios

Questão 5.1:

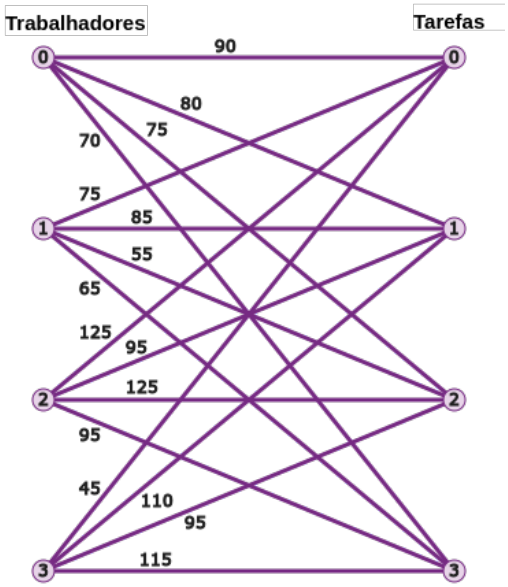
Em uma empresa, um grupo de trabalhadores precisa executar um conjunto de tarefas e, para cada trabalhador e tarefa, há um custo para atribuir o trabalhador à tarefa, representada por uma matriz $Custos_{N \times M}$, no qual N corresponde ao número de trabalhadores e M o número de tarefas

Por ex.:

```
custos[4][4] = {
    {90, 80, 75, 70},
    {35, 85, 55, 65},
    {125, 95, 90, 95},
    {45, 110, 95, 115}
};
```

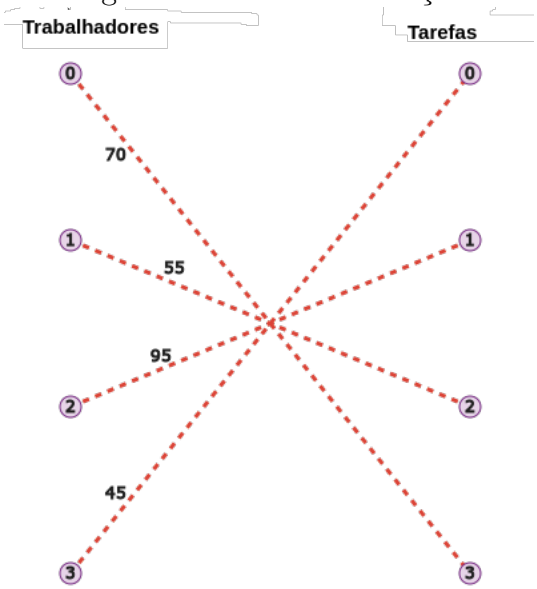
A Figura 1 apresenta uma visualização dessa matriz, no qual existem quatro trabalhadores e quatro tarefas. As arestas (ligações) representam todas as maneiras possíveis de atribuir trabalhadores a tarefas. Os rótulos nas bordas são os custos de designar trabalhadores para tarefas.

Figura 1: Exemplo



Uma atribuição corresponde a um subconjunto das arestas (ligações), em que cada trabalhador tem no máximo uma aresta que sai e não há dois trabalhadores que têm arestas que levam à mesma tarefa. Uma possível atribuição é mostrada na Figura 2 .

Figura 2: Possível atribuição




```
Trabalhador 0 atribuído à tarefa 3. Custo = 70
Trabalhador 1 atribuído à tarefa 2. Custo = 55
Trabalhador 2 atribuído à tarefa 1. Custo = 95
Trabalhador 3 atribuído à tarefa 0. Custo = 45
```

O custo total da atribuição é $70 + 55 + 95 + 45 = 265$.

Como gerente de TI da empresa, sua função é **criar um programa em C, que dada uma matriz *Custos* de entrada, deverá atribuir a cada trabalhador no máximo uma tarefa, sem dois trabalhadores executando a mesma tarefa, minimizando o custo total.**

Quanto menor o custo total gerado pelo algoritmo, ou seja, quanto mais econômica for a atribuição, maior a pontuação da questão

Obs.: O seu programa deverá utilizar ponteiros e structs para representar a solução, e um arquivos para salvar o custo total e as atribuições geradas pelo algoritmo.