Universidade Federal de Viçosa
Centro de Ciências Exatas
Departamento de Informática

2ª PROVA DE PROGRAMAÇÃO I – INF 110 (30 pontos)

NOME_____MAT_____

1. (6 pontos) O código-fonte mostrado no quadro abaixo é compilado sem nenhuma mensagem de erro. No entanto, ao ser executado, o programa exibe uma mensagem de **falha de segmentação**. Leia o código cuidadosamente e então responda as questões que seguem.

```
01 #include <iostream>
02
03 using namespace std;
04
05 void substring(char * a, char b [], int & indice) {
06    for(int k = 0; a != '\0'; k++) {
07         int i = k;
08         int j = 0;
09         while(a[i] == b[j] && a != '\0' && b != '\0') {
10             i++;
11             j++;
12         }
13         if(b == '\0') {
14             indice = i;
15             cout << k << endl;
16         }
17     }
18 }
19
20 int main() {
21     char a[5000000];
22     char b[5000000];
23     cin.getline(a, 5000000);
24     cin.getline(b, 5000000);
25     int c;
26     substring(a, b, c);
27     cout << c << endl;
28     return 0;
29 }
```

a) Quais são os problemas de codificação encontrados na função `substring` que podem causar erros durante a execução do programa? Indique as linhas onde os problemas ocorrem e explique sucintamente como resolve-los.

Os problemas estão nas linhas 6, 9 e 13, quando comparamos arranjos com o caractere \0. A comparação equivocada entre arranjo e caractere afeta os testes booleanos dos comandos for, while e if. A comparação deveria ser feita entre um elemento do arranjo e o caractere, por exemplo: `a[i] != '\0'`.

b) Após corrigir os problemas da função `substring` ainda observa-se a mensagem de **falha de segmentação** durante a execução do programa. Qual problema na função `main` pode causar erro durante a execução do programa? Explique como solucionar o problema.

Os arranjos a e b são possivelmente grandes demais para serem alocados de forma automática. Como a pilha do computador não comporta tais arranjos, o programa termina com um erro de execução. Uma solução seria alocar a e b dinamicamente, visto que tais arranjos seriam alocados no heap, onde consegue-se armazenar mais dados que na pilha.

(não era necessário discutir as diferenças entre pilha e heap, mas apenas apontar o problema e dizer como corrigi-lo)

c) Com todos os problemas resolvidos, faça um rastreio do programa corrigido para a entrada mostrada no quadro abaixo e escreva a saída encontrada no espaço indicado.

Entrada	Saída

`teste de testes`
`te`
	`0`
	`3`
	`9`
	`12`
	`14`

2. (6 pontos) O n-ésimo elemento da sequência de Fibonacci é calculado através da soma dos dois elementos anteriores na série. Considerando que os dois primeiros elementos da série são iguais a um, os primeiros elementos da série são: 1, 1, 2, 3, 5, 8, 13, …

   O programa abaixo deveria calcular o valor do n-ésimo termo da série. Por exemplo, se o valor `n` passado como parâmetro for 6, então o valor retornado deve ser 8. No entanto, a função `fibonacci` possui um erro de implementação. Descreva no espaço abaixo o erro de implementação da função. Não é preciso reescrever o código, mas apenas identificar o erro, explicar como o programa se comporta com o erro e também explicar como corrigi-lo.

```
01 #include <iostream>
02 using namespace std;
03 int fibonacci(int n) {
04     if(n == 1) return 1;
05     return fibonacci(n - 1) + fibonacci(n - 2);
06 }
07
08 int main() {
09     int n;
10     cin >> n;
11     cout << "Fibonacci: " << fibonacci(n) << endl;
12     return 0;
13 }
```

Descreva aqui a sua resposta:

O caso base da função está incorreto. O código deveria retornar 1 para n igual a 1 e 2, e não apenas para n igual a 1. Assim, o código faz chamadas recursivas para valores de n < 0, terminando apenas com um erro de falha de segmentação, quando acabar a memória disponível para armazenar as chamadas recursivas.

3. (6 pontos) Um algoritmo para detectar automaticamente a língua na qual é escrita uma mensagem em uma rede social compara as palavras da mensagem com palavras existentes em dicionários de línguas como Português, Inglês, Alemão, etc. No entanto, a comparação exata das palavras escritas na mensagem com aquelas encontradas no dicionário é dificultada pelo o fato de vários usuários da rede social não escreverem as palavras com a ortografia correta. Para contornar o problema, o algoritmo de detecção de língua não utiliza uma solução exata para a comparação de palavras, mas uma solução aproximada.

   Você deverá implementar uma função em C/C++ para fazer uma comparação aproximada de palavras. A sua função deve receber como entrada duas strings e retornar verdadeiro se as duas strings **concordarem** em pelo menos 50% dos caracteres que as compõem; a função deverá retornar falso, caso contrário. Duas strings **concordam** em relação a um caractere se as duas strings possuem o mesmo caractere em uma dada posição da string.

   Por exemplo, as strings TESTE e TESTES concordam em 5 caracteres, já as strings TTESXES e TESTES concordam em apenas 1 caractere (o caractere T na primeira posição). Para a verificação dos 50% utilizaremos o tamanho da maior string como referência. No primeiro exemplo, TESTES é a maior string com 6 caracteres, e como TESTE e TESTES concordam em 5 caracteres, a função deve retornar verdadeiro, visto que 5/6 é maior ou igual a 0.50. Já no segundo exemplo, a função deve retornar falso, visto que 1/7 é menor que 0.50 (a maior string possui 7 caracteres e o número de caracteres que concordam é 1).

   Não é necessário escrever a função main, apenas a função de comparação das strings.

Escreva aqui a sua solução:

```
bool compara(char * s1, char * s2) {
    int len1 = 0, len2 = 0;
    while(s1[len1] != '\0') len1++; //ou len1 = strlen(s1);
    while(s2[len2] != '\0') len2++; //ou len2 = strlen(s2);

    int maior;
    if(len1 > len2) maior = len1;
    else maior = len2;

    int concorda = 0;
    for(int i = 0; s1[i] != '\0' && s2[i] != '\0'; i++)
        if(s1[i] == s2[i])
            concorda++;
    if(1.0*concorda/maior >= 0.5)
        return true;
    return false;
}
```

4. (6 pontos) No jogo de Jokenpô dois jogadores revelam simultaneamente, através de sinais com as mãos, suas escolhas dentre as opções: Pedra, Papel e Tesoura. A Pedra empata com Pedra, perde de Papel e ganha de Tesoura; Papel ganha de Pedra, empata com Papel e perde de Tesoura; Tesoura perde de Pedra, ganha de Papel e empata com Tesoura.

   A matriz de ganhos do jogo é mostrada na figura abaixo, onde os valores de +1, 0 e -1, representam os valores de vitória, empate e derrota do jogador linha (Jogador 1) versus o jogador coluna (Jogador 2). Por exemplo, se o Jogador 1 jogar Pedra e o Jogador 2 jogar Papel, o primeiro receberá uma recompensa de -1, representando a derrota no jogo.

	Pedra	**Papel**	**Tesoura**
Pedra	0	-1	+1
Papel	+1	0	-1
Tesoura	-1	+1	0

   A estratégia de melhor resposta do Jogador 1 para uma estratégia fixa do adversário, Jogador 2, é aquela que dará ao Jogador 1 a maior recompensa possível para uma dada coluna da matriz. Escreva uma função em C/C++ que recebe como entrada a matriz de ganhos de um jogo e o índice da estratégia do Jogador 2, e retorna o índice da linha representando a estratégia de melhor resposta para a estratégia do Jogador 2.

   Por exemplo, ao fornecermos a matriz de ganhos de Jokenpô mostrada acima com a estratégia de índice 0 (Pedra) para o Jogador 2, a função deverá retornar o valor 1, representando a estratégia Papel para o Jogador 1, pois na coluna 0 a entrada correspondente a Papel possui o maior valor. Se a estratégia do Jogador 2 fosse Tesoura, a função deveria retornar 0, visto que Pedra é a melhor resposta para Tesoura.

   Repare que não se sabe a priori qual matriz de ganhos será passada como parâmetro, por isso o seu programa deve funcionar **para qualquer matriz**. Pode-se assumir que existirá apenas uma estratégia de melhor resposta para a estratégia do Jogador 2. A sua solução não pode assumir que os valores de recompensa serão sempre iguais a +1, 0 e -1. No entanto, pode-se assumir que os valores serão inteiros e que podem ser armazenados em variáveis do tipo int.

Escreva aqui a sua solução:

```
int melhor_resposta(int ** matriz, int linhas, int jogador2) {
    int recompensa = matriz[0][jogador2];
    int indice = 0;
    for(int i = 1; i < linhas; i++) {
        if(matriz[i][jogador2] > recompensa) {
            indice = i;
            recompensa = matriz[i][jogador2];
        }
    }
    return indice;
}
```

5. (6 pontos) Crie um tipo heterogêneo de dados chamado `Retangulo` contendo dois campos do tipo inteiro, um para determinar o comprimento e outro para determinar a altura de um retângulo. Então escreva um programa que possibilite o usuário digitar uma coleção de retângulos, onde o usuário irá escolher quantos retângulos serão digitados. O seu programa deve então ordenar os retângulos digitados.

   O critério de ordenação deve ser o seguinte: retângulos com menor comprimento deverão vir antes dos retângulos com maior comprimento. Caso haja empate entre o comprimento de dois retângulos, o valor da altura deve ser utilizado como critério de desempate, fazendo com que retângulos com menor altura sejam posicionados antes dos retângulos com maior altura. Não serão digitados retângulos com o mesmo comprimento e a mesma altura.

   Utilize o algoritmo da bolha para ordenar os retângulos; veja o pseudocódigo abaixo:

```
Para i = n-1 até 0 faça:
      Para j = 0 até i faça:
            Se o itens nos índices j e j+1 estiverem fora de ordem
                  Troque os itens j e j+1 de posição.
```

   Nos exemplos abaixo o usuário primeiro digita a quantidade N de retângulos e depois fornece N pares de inteiros, onde cada par especifica o comprimento e a altura de um retângulo. O programa então ordena os retângulos, imprimindo-os em seguida na tela.

Exemplos

Entrada	Saída
3 3 4 4 2 3 1	3 1 3 4 4 2

```
struct Retangulo {
    int c; int a;
};
void bolha(Retangulo rets [], int n) {
    for(int i = n-1; i > 0; i--)
        for(int j = 0; j < i; j++)
          if(rets[j].c > rets[j+1].c ||
           (rets[j].c == rets[j+1].c && rets[j].a > rets[j+1].a)){
                Retangulo aux = rets[j];
                rets[j] = rets[j+1];
                rets[j+1] = aux;
            }
}
int main () {
    int n; cin >> n;
    Retangulo rets[n];
    for(int i = 0; i < n; i++)
        cin >> rets[i].c >> rets[i].a;
    bolha(rets, n);
    for(int i = 0; i < n; i++)
        cout << rets[i].c << " " << rets[i].a << endl;
}
```