Universidade do Estado Santa Catarina - UDESC
Departamento de Ciência da Computação - DCC
IV - Maratona de Programação – 2007

Lucas, Claudio e Rosso

6 de junho de 2007

Problema A: Computador de Palavras

Arquivo fonte: String.c String.cpp String.java

A DataAzul comprou recentemente o seu mais novo computador, um computador de processamento de palavras chamado de X9091. Espera-se que este tenha alguma valor/utilidade para criptografia e áreas afins. (se espalham boatos que os chineses estão trabalhando em um clone que corrija tal codificação criptográfica, mas nós ignoraremos tal rumor). Este computador aceitará entradas de palavras e produzirá como saída outras palavras, dependendo do programa carregado dentro dele naquele momento. O chip da CPU é de uma última geração com tecnologia RISC. Contudo, este tem somente três instruções na transformação de palavras:

D Delete ou exclua um caracter numa posição em particular.

I Insira um caracter em uma posição em particular.

C Mude o caracter de uma posição em particular por um outro caracter diferente. Isto é, troque de caracter em uma dada posição.

Os programas escritos para esta máquina estão em um formato de código de máquina onde cada instrução tem o formato **ZXdd—**, onde **Z** representa o código para uma instrução (D, I ou C), **X** é um caráter e o **dd** representa um número de dois dígitos. Um programa é terminado por uma instrução especial da parada que consiste na letra 'E'. Observe que cada instrução trabalha sobre a palavra na memória cada vez que a instrução é executada.

Para ver como tudo isso funciona considere o seguinte exemplo. Deseja-se transformar a palavra 'abcde' para palavra 'bcgfe'. Isto poderia ser obtido por uma série de comandos da mudança (C), mas este não seria minimal. O seguinte programa é melhor:

```
        abcde
Da01    bcde    % observe o 'a' é necessário porquê ele é verificado pelo hardware
Cg03    bcge
If04    bcgfe
E       bcgfe   % Término o programa
```

Escreva um programa que leia duas palavras (uma palavra de entrada e uma palavra de saída ou alvo) e produza um programa minimal em X9091 necessário transformar a palavra de entrada na palavra de saída. Desde que podem haver múltiplas soluções, somente uma deve ser produzida. Qualquer solução que satisfaça estes critérios será aceita.

A entrada consistirá em uma série das linhas, cada linha que contendo duas palavras separadas por exatamente um espaço. As palavras tem no máximo 20 caráteres, todos minúsculos. O arquivo será finalizado por uma linha contendo um único #.

A saída consistirá em uma série das linhas, uma para cada linha da entrada. Cada linha consiste em um programa na línguagem X9091.

Exemplo de entrada

```
abcde bcgfe
#
```

Exemplo de saída

```
Da01Cg03If04E
```

Problema B: Os Successores da Sultana

Arquivo fonte: Sultana.c Sultana.cpp Sultana.java

A sultana de Nubia não tem filhos, assim ela decidiu que seu país será divido em até K regiões após a sua morte, e cada parte será herdada por aquele que melhor executar um dado teste. É possível que um dado indivíduo possa herdar mais de uma destas partes, ou ainda, todo reino. Para assegurar que somente pessoas altamente inteligentes se transformem eventualmente em um de seus sucessores, a sultana planejou um teste engenhoso. No salão principal de sua morada, repleto de fontes de água, com um delicado incenso no ambiente, foram colocados K tabuleiros de xadrez. Cada tabuleiro tem números escritos em cada quadrado na escala 1 a 99 e são fornecidas 8 rainhas do jogo de xadrez. A tarefa a ser enfrentada por cada sucessor em potencial é colocar as 8 rainhas no tabuleiro de xadrez, de tal maneira que nenhuma rainha ameaçe a outra, e de modo que os números nos quadrados escolhidos para estas rainhas, totalize na maior soma destes números, o qual deve ser pelo menos tão alto quanto um valor já escolhido pela sultana. (Para aqueles que desconhecem as regras do xadrez, isto implica dizer que em uma dada casa onde uma rainha se encontre, naquela linha e coluna desta casa e suas diagonais, não podem conter mais nenhuma rainha.)

Escrever um programa que leia o número K de tabuleiros e em seguida os seus números ou configurações de cada tabuleiro, sequencialmente. Determine as somas mais elevadas para cada tabuleiro sob estas circunstâncias. (Você sabe que a sultana é uma boa jogadora de xadrez e uma boa matemática e você suspeita que o score alcançado por ela é o melhor a ser atingido.)

A entrada consiste em K (o número das tabuleiros), em uma linha, seguido por K conjuntos de 64 números, onde cada conjunto consiste em oito linhas e oito números por linha. Ou seja, a descrição do tabuleiro. Cada número será um inteiro positivo menor de 100. Nunca haverá mais de 20 tabuleiros.

A saída consistirá em K números, representando os seus K scores, onde cada score é impresso por linha e justificado à direita num campo de 5 caráteres.

Exemplo de entrada

```
1
 1  2  3  4  5  6  7  8
 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
48 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
```

Exemplo de saída

```
  260
```

No exemplo acima, há apenas um tabuleiro. Se tivéssemos dois, este o número K seria igual a 2, e a descrição do próximo tabuleiro começaria logo a subsequente a última linha do primeiro tabuleiro.

Problema C: Theseu e o Minotauro

Arquivo fonte: Theseu.c Theseu.cpp Theseu.java

Aqueles de vocês com educação clássica devem lembrar da lenda de Theseu e o Minotauro. Esta a fábula envolvendo um mostro com cabeça de touro, um labirinto subterrâneo com pequenas passagens parecidas, amores perdidos de donzelas e bolas de seda. Na linha educacional desta competição, iremos agora revelar a verdadeira estória.

O labirinto era na verdade uma série de cavernas conectadas por passagens diretas, algumas das quais só podiam ser atravessadas em uma direção. Para encurralar o Minotauro, Theseu levou escondido consigo uma grande quantidade de velas para dentro do Labirinto, pois ele havia descoberto que o Minotauro tinha medo da luz. Theseu caminhou a esmo até ouvir o Minotauro aproximando-se pelo túnel. Neste momento ele acendeu uma vela e partiu em perseguição. O Minotauro bateu em retirada para dentro da caverna da qual havia saído e fugiu por outra passagem. Theseu o seguiu lentamente, até que encontrou a k-ésima caverna desde que ele acendeu a vela. Aqui ele teve tempo suficiente para colocar a vela acesa no meio da caverna, acender outra vela naquela e continuar a perseguição. Com o progresso da perseguição, uma vela era deixada em cada k-ésima caverna que era atravessada, assim limitando o movimento do Minotauro. Sempre que o Minotauro entrava numa caverna ele devia checar as saídas em uma ordem específica, fugindo por aquela que não levasse diretamente a uma caverna iluminada. (Lembre-se que como Theseu carregava uma vela acesa, o Minotauro nunca saia da caverna pelo túnel usado para entrar nela). Eventualmente o Minotauro ficou encurralado, possibilitando Theseu derrotá-lo.

Considerando os Labirintos a seguir como exemplo, onde no caso o Minotauro verifica as saídas das cavernas em ordem alfabética:

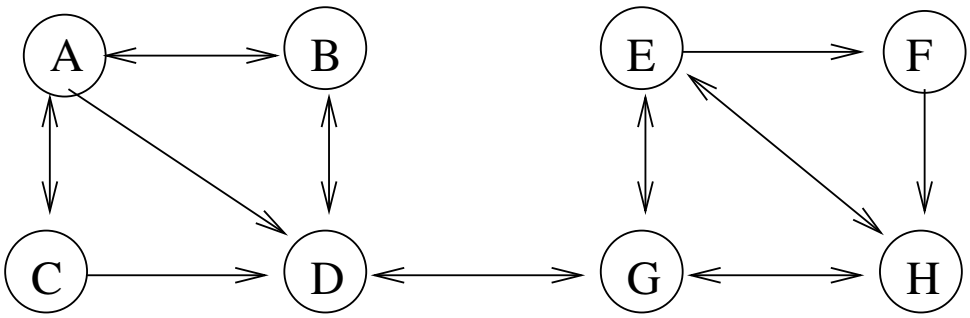

Figura 1: Uma caverna em ordem alfabética

Assumindo que Theseu está na caverna C quando ouve o Minotauro se aproximandon vindo de A, e para este cenário o valor de K é 3. Ele acende a vela e começa a caçada, perseguindo o Minotauro através de A,B, D(deixa uma vela), G, E, F(outra vela), H, E, G(outra), H, E (encurralado).

Escreva um programa que simulará a perseguição de Theseu ao Minotauro. A descrição do labirinto vai identificar cada caverna por um caractere maiúsculo e irá listar as cavernas alcançáveis a partir daquela na ordem que o Minotauro irá tentar, seguida dos identificadores para as cavernas as quais o Minotauro e Theseu estavam quando o contato foi feito pela primeira vez, seguido pelo valor

de k. O arquivo de entrada consistirá de uma série de linhas. Cada linha descreverá um cenário no formato mostrado abaixo(que descreve o exemplo acima). Nenhuma linha poderá conter mais de 255 caracteres. O arquivo será finalizado por uma linha com apenas o caractere #.

O arquivo de saída irá consistir de uma linha para cada labirinto. Cada linha irá identificar as cavernas iluminadas, na ordem na qual as velas foram deixadas, e a caverna na qual o Minotauro está encurraldo, seguindo o formato mostrado abaixo.

Exemplo de entrada

```
A:BCD;B:AD;D:BG;F:H;G:DEH;E:FGH;H:EG;C:AD. A C 3
#
```

Exemplo de saída

```
D F G /E
```

Problema D: Dobradura de Papel

Arquivo fonte: Dobradura.c Dobradura.cpp Dobradura.java

Se uma grande folha de papel é dobrada na metade, e então na metade outra vez e assim sucessivamente, com todas as dobras paralelas então, existirá uma série de vincos (marcas de dobra) paralelas. Algumas apontando para cima, outras para baixo, dividindo o papel em fracões do comprimento original. Se o papel for aberto apenas "*meio caminho*" em cada dobra, então dobra formará um ângulo de 90 graus. Então, (vista pela borda) ela forma uma "curva dragão". Por exemplo, se quatro dobras sucessivas forem feitas a curva a seguir será vista(note que não cruza ela mesma, mas dois cantos se tocam):

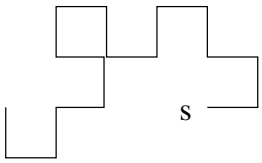

Figura 2: Desdobrando o papel

Escreva um programa para desenhar a curva que aparece após N dobras. A especificação exata da curva é a seguinte: o papel inicialmente é plano, com a "*borda inicial*" à esquerda, se olhado de cima. A metade direita é dobrada de forma que cobre a metade esquerda. Então, a metade direita da nova "*folha dupla*" é dobrado sobre a metade esquerda, para formar uma folha de 4 camadas, e assim por diante, até N dobraduras. Então, cada dobra é aberta de uma dobra de 180graus para uma dobra de 90graus. Finalmente o papel é olhado pela borda para vermos a "curva dragão". Nesta vista, a única parte não modificada da folha original é o pedaço contendo a "borda inicial" e esta parte está na horizontal, com a "*borda inicial*" na esquerda. Isto define a curva de maneira única. Na figura acima, a "*borda inicial*" é a terminação esquerda da parte horizontal da curva no canto inferior direito(marcada com o 's' de Start em Inglês). A partes horizontais deverão ser mostradas com caractere sublinhado("_"), e as partes verticais deverão ser mostradas com o caráter "|".

O arquivo de entrada consistirá de uma série de linhas, cada uma com um único número N ($1 \leq N \leq 13$). O final das entradas será marcado por uma linha contendo um Zero.

O arquivo de saída consistirá de uma série de “curvas dragão", uma para cada valor de N na entrada. Sua figura deve estar deslocada o mais a esquerda e acima possível. Note que para N grandes, a figura terá mais de 80 caracteres de largura, o que parecerá confuso na tela. O padrão gerado para cada número diferente de dobraduras terminará com um caractere ‘^’.

Exemplos de entradas

```
2
4
1
0
```

Exemplos de saídas

```
|_
 _|
^

    _   _
   |_|_|  |_
    _|     _|
|_|
^

_|
^
```

Problema E: Detecção de Desmatamento

Arquivo fonte: Desmata.c Desmata.cpp Desmata.java

Uma matriz de A linhas e B colunas, resultado do processamento de uma imagem de satélite, é formada por inteiros sem sinal de 3 bits. Ou seja cada elemento, na linha x e coluna y, pode assumir um entre oito valores diferentes, associado à região em que faz parte na imagem:

0 - Cidade
1 - Nuvem
2 - Mata
3 - Rio/Lago/Mar
4 - Neve
5 - Montanha
6 - Deserto
7 - Outra

Um órgão do governo de Santa Catarina pretende, a partir de duas destas matrizes pré-processadas a partir de imagens de uma mesma região de Joinville (uma de 1987 e outra de 2007 com coordenadas geográficas idênticas), verificar todos os pontos de desmatamento (d) e de reflorestamento (r) e,

ao final, ter uma índice de impacto ambiental. Tal índice é obtido pela *quantidade de elementos de desmatamento* dividido pela *quantidade de elementos de reflorestamento*, com duas casas decimais. Observe que há, portanto, interesse em verificar a presença ou ausência de elementos da matriz com valor igual ou diferente a 2 de acordo com a lista de significados descrita anteriormente. Considera-se que a mudança de 2 para qualquer outro valor, representa desmatamento. O contrário também é válido, qualquer valor que mude para 2 representa reflorestamento.

Entrada

A entrada é consistituída apenas por números inteiros divididos da seguinte forma: (a) linha inicial com dois valores, sendo o primeiro referente ao número de linhas (A) da matriz e o segundo, o número de colunas (B); (b) linha seguinte com o ano de aquisição da primeira imagem de satélite; (c) A linhas seguintes com os valores da primeira matriz; (d) linha seguinte com o ano de aquisição da segunda imagem de satélite; (e) A linhas seguintes com os valores da segunda matriz. O tamanho desta matriz é limitado 300 por 300. Não há carácter para o final da entrada e esta é de apenas um mapa único. Ou seja, como o do exemplo a seguir.

Saída

A saída é construída nesta seqüência: (a) primeira linha com dois valores indicando número de linhas (A) e de colunas (B); (b) segunda linha com primeiro ano e segundo ano; (c) terceira linha em diante com uma matriz de caracteres, no qual cada elemento pode ser:

- **.** se não houve alteração
- **r** se houve reflorestamento
- **d** se houve desmatamento

E, por fim, (d) última linha contendo o índice de impacto ambiental pela cardinalidade(r)/cardinalidade(d) com arredondamento em duas casas decimais.

Exemplo de entrada

```
3 5
1987
7 7 0 0 2
7 3 2 0 0
2 2 3 0 2
2007
7 2 0 0 0
7 3 0 0 0
2 2 3 2 0
```

Exemplo de saída

```
3 5
1987 2007
.r..d
```

```
..d..
...rd
0.67
```