Universidade Estadual de Campinas – Instituto de Computação
Segundo Semestre de 2015
MC202 – Estrutura de Dados – Turmas G e H
Professor Lehilton Pedrosa

Primeira lista de exercícios

1. Veja o código:

```
void f(int a, int b) {
        int c = a;
        a = b;
        b = c;
}
void g(int *c, int n) {
        c = malloc(sizeof(int) * n);
}
void h(int *h) {
        *h = 100;
}
int main() {
        int a = 1, b = 2, c= 3;
        int **d;
        d = malloc(sizeof(int*));
        f(a, b);
        g(d[1], 1);
        h(*d);
        free(d);
        printf("%d, %d, %d", a, c, d[1][1]);
        return 0;
}
```

a) É possível determinar a saída do programa? Qual?
b) Existe *memory leak*?
c) Modifique para que toda memória alocada seja liberada e a saída seja a esperada: 2, 3, 100.

2. Crie um tipo de dados abstrato "número complexo". As seguintes operações devem estar disponíveis:
a) inicializar número
b) imprimir número
c) soma/multiplicar/subtrair números
d) potência e divisão por números (deve imprimir uma mensagem de erro sempre que a operação não estiver bem definida)
e) comparar números

3. Crie um tipo abstrato de dados "sequência de números complexos". As seguintes operações devem estar disponíveis (com exceção das duas primeiras, nenhuma operação deve alterar o conteúdo das sequências passadas como parâmetros). Crie a interface e implemente algumas das funções
a) iniciar sequência
b) destruir sequência
c) prefixo/sufixo de tamanho n
d) concatenar sequências
e) comparar sequências

f) criar sequência das séries, isso é, se a sequencia é $(a_1, a_2, \ldots, a_n)$ a sequência de séries é $(a_1, a_1 + a_2, \ldots, a_1 + a_2 + \cdots + a_n)$

4. Notação O:

a) Escreva as funções a seguir usando a notação O adequada mais simples:

- $n^5 + 100000$;
- $\log_{35}(n^n)$;
- $2^{n-0,1}$;
- $2^{\log n^2}$;
- $1000^{1000n} + n^n$.

b) Prove ou desprove as afirmações:

- $2^n = O(2^{n-0,001})$;
- $2^n = O(2^{n(1-0,001)})$;
- $n^2 = O(n^{1,999})$.

5. (longo) Em um banco, há apenas um caixa e todos devem ser atendidos por ordem de chegada. No entanto, pessoas idosas têm prioridade e passam para o início da fila. Considerando que em vários dias o número de idosos é muito grande, foi estipulada a seguinte regra:

1. uma pessoa é atendida na ordem de chegada
2. no máximo 2 idosos podem passar na frente de uma pessoa que não é idosa

Escreva um programa que leia uma sequência de linhas, onde cada linha contém a informação da ordem de chegada e categoria do cliente e imprima a ordem de atendimento (considere que todos chegaram antes de começar o atendimento).

Ex. de entrada:

```
1 geral
2 geral
3 idoso
4 idoso
5 idoso
6 geral
7 idoso
```

Ex. de saída:

```
3 4 1 2 5 7 6
```

6. (Tanenbaum) Escreva um algoritmo para determinar se uma string de caracteres de entrada é da forma:

$$xCy$$

onde x é uma string consistindo de letras 'A' e 'B' e y é o inverso de x (isto é, se x é a string "ABABBA", então y deve equivaler a "ABBABA"). Em cada ponto, você só poderá ler o próximo caractere da string.

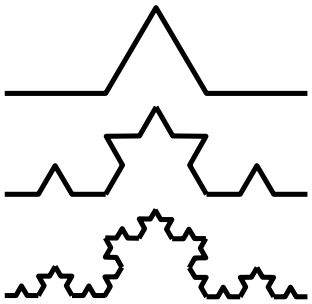
7. Implemente um tipo abstrato de dados "pilha de inteiros com máximo", eficientemente, com as seguintes operações: empilhar inteiro, desempilhar inteiro, obter valor máximo. *Cada operação deve ter tempo de execução constante!*

8. Calcule o número de movimentos necessários para o problema das torres de Hanói com n discos.

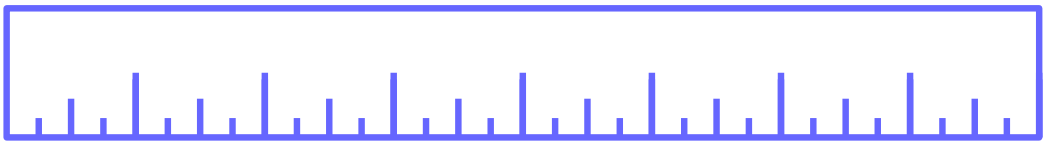
9. Um programa LOGO é uma sequência de instruções que se baseiam em um cabeçote de desenho (chamado de tartaruga) numa certa posição atual e orientação em uma tela de desenho e nos seguintes comandos:

- **desenhar**(x): desenha e anda em linha reta a partir da posição atual de tamanho x pixels;
- **girar**(x): gira o cabeçote em um ângulo de x graus;

Uma estrela de Koch é um fractal como na imagem a seguir, nos níveis 1, 2 e 3, respectivamente. Escreva uma função que escreva uma sequência de comandos LOGO para desenhar a estrela de Koch de nível l de largura w.



10. Escreva uma função recursiva que escreva um programa LOGO para desenhar uma régua, com marcadores de 1cm, 0,5cm, 0,25cm proporcionais, como na figura (considere que cada cm equivale a 100 pixels). Depois, tente implementar uma versão iterativa do problema.



11. Como é possível implementar o *Merge-Sort* de maneira iterativa (observar o exercício 10 pode ser útil)? Esboce ou implemente o *Merge-Sort* como um algoritmo iterativo.

12. (longo) Uma turma de 5 amigos decide fazer um amigo-oculto um pouco diferente. Ao invés de um só presente, cada pessoa vai dar dois presentes para duas pessoas diferentes. Claro, para que ninguém fique prejudicado, cada um também vai receber presentes de duas pessoas diferentes.

a) Escreva um programa para obter uma distribuição de presentes possível.

b) Como você faria para fazer essa distribuição e preservar o segredo?

c) Para deixar o problema ainda mais complicado, no ano seguinte, eles decidiram adicionar ainda mais duas regras:

- cada pessoa escreve uma lista de pessoas de quem não quer receber presente
- cada pessoa escreve uma lista de pessoas para quem não quer dar presente

Como nem sempre vai ser possível satisfazer todo mundo, escreva um programa que minimize o número de pessoas insatisfeitas. (Dica: uma maneira de resolver esse problema é testar todas as soluções)