MAC 122 – Princípios de Desenvolvimento de Algoritmos

BCC – Prova de Recuperação – 16 de fevereiro de 1998

NOME DO ALUNO :
NUSP :
ASSINATURA:

INSTRUÇÕES

1. Preencha o cabeçalho acima.
2. Verifique se sua prova está completa. Esta prova deve ter 5 folhas (contando esta capa), com o enunciado de 4 questões.
3. Você pode fazer essa prova a lápis.
4. A prova deve ser resolvida sem consulta a apontamentos, cadernos, livros ou colegas.

DURAÇÃO DA PROVA: 3 horas

	Nota
Questão 1	
Questão 2	
Questão 3	
Questão 4	
TOTAL	

B O A   P R O V A

1. (valor 2.5 pontos)
   Podemos representar uma seqüência de operações sobre pilhas (inicialmente vazia) através de uma seqüência de "D" e "E", sendo que o primeiro indica que ocorreu um desempilha e o segundo indica que ocorreu um empilha. Dizemos que uma seqüência de "D" e "E" é bem formada se nenhum desempilha é feito sobre uma pilha vazia e no final da seqüência a pilha deve estar vazia.

   Por exemplo, a seqüência EEEDDEEDDD é bem formada. Já a seqüência EEDDED**D**EEE não é bem formada, pois o quarto desempilha (indicado em negrito) é feito em uma pilha vazia. A seqüência EED também não é bem formada, pois a pilha não está vazia no final.

   Faça um programa utilizando pilhas que verifica se uma seqüência de "D" e "E" é bem formada.

2. (valor 2.5 pontos)
   Faça uma função recursiva que recebe um vetor V com n elementos, e um elemento x e devolve o número de elementos do vetor menores e maiores que x. Faça uma análise de quantas comparações entre x e elementos do vetor são feitas no melhor e pior caso.

3. (valor 0.8+0.8+0.9 pontos)

a. Simule a execução do algoritmo `Separa` do Quicksort para o vetor abaixo, indicando as comparações e trocas.

12	3	24	10	6	67	8	5	45	46	78	9	14	7	3	1	14

b. Simule a execução do algoritmo `Heapfica` do Heapsort para o vetor abaixo, indicando as comparações e trocas.

12	3	24	10	6	67	8	5	45	46	78	9	14	7	3	1	14

c. Justifique rigorosamente por que o Heapfica tem complexidade $O(n)$.

4. (valor 1+0.5+1.0 pontos)
Considere o seguinte algoritmo para verificar se um número $n \geq 2$ é primo.

```
primo = true;
for (int i = 2; i < n && primo; i++)
    if (n % i == 0)            (*)
       primo = false;
```

a. Se o número de bits da representação binária de n é k, quantas vezes, em função de k, a comparação (*) é executada?

b. Mostre que basta procurarmos divisores até $\sqrt{n}$.

c. Qual a resposta para o item (a.) se trocamos a condição $i < n$ por $i \leq \sqrt{n}$?