Gabarito da Lista de Algoritmos e Estrutura de Dados

Questão 1)

Verificação por métodos invariantes:

propriedade invariante:

A variável x corresponde ao valor da exponencial a^i, em que i é o número de iterações do loop *while* (0 antes do loop).

Inicialização:

A variável x é inicializada com $1 = a^0$, sendo válido a propriedade acima.

Manutenção:

No final de cada loop $x = x_0 * a = a^{(i_0+1)} = a^i$ sendo i_0 o número de iterações do loop sem contar com a corrente (ou o número de multiplicações anteriores por a) e x_0 o valor de x antes do loop.

Término:

O processo finaliza quando b, que é decrementado em cada iteração, chega em 0. Em outras palavras, o processo encerra depois de b iterações, e de acordo com a propriedade, x encerra com o valor de a^b, o que é o esperado para a exponenciação natural.

Questão 2)

Verificação por métodos invariantes:

propriedade invariante:

A variável x corresponde ao valor do produto $x * i$, em que i é o número de iterações do loop *while* (0 antes do loop).

Inicialização:

A variável x é inicializada com $0 = x * 0$, sendo válido a propriedade acima.

Manutenção:

A cada iteração, x é somado ao argumento a uma vez, ou seja , sendo i_0 o número de iterações do loop sem contar com a corrente (ou somas anteriores de a), e modo que a propriedade é mantida.

No final de cada loop $x = x_0 + a = a * (i_0 + 1) = a * i$ sendo i_0 o número de iterações do loop sem contar com a corrente (ou o número de soma anteriores com a) e x_0 o valor de x antes do loop.

Término:

O processo finaliza quando i, que é incrementado em cada iteração, chega em a. Em outras palavras, o processo encerra depois de a iterações, e de acordo com a propriedade, x encerra com o valor de $a * a = a^2$, o que é o esperado.

Questão 3)

Verificação por métodos invariantes:

propriedade invariante:

A variável x corresponde ao valor do fatorial $i!$, em que i é o número da atual iteração do loop *while* (começando em 1).

Inicialização:

A variável x é inicializada $1 = 0!$, sendo válido a propriedade acima.

Manutenção:

A cada iteração, x é multiplicado por i, ou seja $x = x * i = i_0!* i = i!$, sendo i_0 o número de iterações passadas do loop, sem contar a presente, de modo que a propriedade é mantida.

Término:

O processo finaliza quando i, que é incrementado em cada iteração, chega em a. Em outras palavras, o processo encerra depois de a iterações, e de acordo com a propriedade, x encerra com o valor de $a!$, o que é o esperado.

Questão 4)

Algoritmo correto:

```
unsigned produto_corrigido(unsigned a, unsigned b)
{
   if (b == 0)
     return 0;
  x = a;
  resto = 0;
  while (b > 1)
  {
    if (b%2) resto = resto + x;
    x = x * 2;
    b = b / 2;
  }
  return x+resto;
}
```

ou alternativamente na forma recursiva:

```
unsigned produto_recursivo( unsigned a, unsigned b)
{
  if ( b == 0 )
    return 0;
  if ( b == 1 )
    return a;
  if ( b % 2 )
    return a + produto_corrigido ( 2*a,b/2 );
  else
    return produto_corrigido ( 2*a, b/2 );
}
```

A análise será feita a partir da forma direta, embora ambas formas são equivalentes, com a diferença que na forma direta a pilha (vide implementação da recursão em compiladores) é gerenciada manualmente usando a variável resto.

Explicação do algoritmo:

Esse algoritmo parte do pressuposto que multiplicar uma parcela por 2 e dividir a outra por 2 mantém o resultado na multiplicação. Por exemplo $2 * 12 = 4 * 6 = 8 * 3$ ou genericamente $a * b = (a * 2)(b / 2)$

Para o caso de b ímpar (não divisível por 2), para manter o valor do produto constante, é necessário acrescentar o resto da divisão ao total (que segundo a conta abaixo equivale à constante a, ou o número sendo multiplicado por 2 pelo algoritmo):
$a * b = (a * 2)(b / 2) = (a * 2)[(b - 1)/2 + 1/2] = (a * 2) * (b - 1)/ 2 + a$
em que (b-1)/2 = divisão inteira de um número ímpar (como ocorre em divisão de *int* em linguagem C).

Exemplo:
$3 * 5 = 6 * 2 + 3$
Exemplo completo:
$3 * 15 = 6 * 7 + 3 = 12 * 3 + 6 + 3 = 24 * 1 + 12 + 6 + 3 = 24 + 21 = 45$

propriedade invariante:

Como explicado acima, a propriedade invariante é que o produto x*b+resto é sempre constante no final do loop e igual ao resultado correto da multiplicação.

Inicialização:

A variável $b = 0$ é considerada um caso especial, e o resultado correto é retornado imediatamente. Para o caso de $b > 0$, x é inicializado como a e o resto como 0, de modo que a propriedade invariante $x * b + \mathrm{resto} = a * b + 0 = a * b$ que não viola a propriedade invariante.

Manutenção:

A cada iteração, 2 situações podem ocorrer como visto na explicação:

b par: $b = b/2$, $x = x * 2$, $\mathrm{resto} = \mathrm{resto}$: mantém $x * b + \mathrm{resto}$ constante, já que como visto $x * b = (x * 2)(b/2)$, e logicamente $x * b + \mathrm{resto} = (x * 2)(b/2) + \mathrm{resto}$

b ímpar: $b = (b - 1)/2$, $x = x * 2$, $\mathrm{resto} = \mathrm{resto} + x$ (nota: ver sobre divisão inteira em C acima): mantém $x * b + \mathrm{resto}$ constante, já que como visto
$x * b = (x * 2) * (b - 1)/2 + x$, e logicamente
$x * b + \mathrm{resto} = (x * 2)(b/2) + (\mathrm{resto} + x)$;

Término:

O algoritmo encerra quando $b = 1$ (nota: sempre através de divisões inteiras sucessivas o número chega a 1, a menos que b seja 0, o que é tratado de forma especial, ou b já seja 1, de modo que o loop encerra antes da primeira iteração).
Nessa situação, $x * b + \mathrm{resto} = x * 1 + \mathrm{resto} = x + \mathrm{resto} = a * b$, de modo que o resultado correto da multiplicação é retornado.

Questão 5)

a)
Uma possível implementação é:

```
void selection_sort(int *array, unsigned size)
{
   int *min;
   int aux;
   unsigned i, j;
   for (i = 0; i < size-1; i++)
   {
      min = &array[i];
      // procurar o menor elemento pelo resto do array
      for (j = i; j < size; j++)
      {
        if (array[j] < *min)
            min = &array[j];
      }
      // fazer a troca
      aux = *min;
      *min = array[i];
      array[i] = aux;
   }
}
```

b)
Loop interior (achar o menor valor da lista)
propriedade invariante:

A variável min sempre possui o menor valor entre os elementos do array nas posições i até j.

Inicialização:

A variável min é inicializada como o valor na posição i, a solução correta para o menor valor de até j=i.

Manutenção:

A cada iteração, se o valor em j for menor que o valor mínimo guardado em min das posições i até j-1, ele é guardado em min, de modo que a variável sempre possui o menor valor de i à j.

Término:

O término ocorre quando foi vasculhado todo o array, ou seja, quando j=size-1.

Loop externo (o loop que executa as trocas)

propriedade invariante:

Após a iteração o array está ordenado das posições 0 à i e o maior valor nessa faixa (na posição i) é menor do que todos os valores nas posições seguintes.

Inicialização:

A variável i é inicializada com 0 para apontar para o primeiro elemento do array. A propriedade invariante é válida já que o array está ordenada para elementos antes de i = 0 (um grupo de zero elementos no entanto).

Manutenção:

A cada iteração, o elemento de posição j é trocado com o menor elemento de j+1 até o fim da lista, de modo que o elemento em j é sempre menor do que todos os elementos seguintes ao fim da iteração (e maior que os elementos das posições 0 até j-1).

Término:

O término ocorre quando a iteração foi executada para o penúltimo elemento da lista ($i = \mathrm{size} - 2$). Nesse momento, o array está ordenado das posições 0 à $i = \mathrm{size} - 2$ (e, como explicado na letra c desse exercício, está ordenado no array todo).

c)

A propriedade invariante do loop externo garante que o valor em i é menor do que todos os valores restantes no arranjo. Desse modo é garantido que o último elemento é o maior em valor quando i é o penúltimo elemento, e portanto, o arranjo está completamente ordenada.

Questão 6)

a)
Loop interior:

propriedade invariante:

O elemento em A[j-1] é sempre o menor entre os elementos das posições j-1 até o final do array na posição n-1.

Nota-se também que o array é uma permutação válida pois as únicas operações feitas são trocas de dois elementos (por meio de uma variável auxiliar aux).

Inicialização:

Antes da primeira iteração, j é inicializado como n-1, apontando para o último elemento do array. A propriedade invariante é válida pois, antes do loop, o elemento em A[j=n-1] é o menor dentre os elementos de posição maior ou igual a n-1 (por ser o único elemento desse grupo).

Manutenção:

Ao final de cada iteração, se o elemento em j-1 for maior que o elemento em j, os dois são trocados de modo que o elemento em j-1 passa a ser (ou continua a ser) o menor elemento do grupo entre as posições j-1 até n-1.

Término:

O término ocorre quandoj=i+1, e o menor elemento de todo grupo seja A[i]

b)
Loop externo (similar ao loop externo da questão 5):

propriedade invariante:

Após a iteração o array está ordenado das posições 0 à i e o maior valor (em i) é menor do que todos os valores nas posições seguintes.

Inicialização:

A variável i é inicializada em 0 para apontar para o primeiro elemento do array. A propriedade invariante é válida como o array está ordenado para elementos antes de $i = 0$ (um grupo de zero elementos no entanto).

Manutenção:

A cada iteração, o elemento em i passa a ser o menor elemento do sub-array entre as posições i e $n - 1$ (fim do array), mantendo a propriedade de que todos os elementos de 0 à i estão ordenados e são os menores valores do array.

Término:

O término ocorre quando a troca foi executado para o penúltimo elemento da lista, garantido que (de forma similar ao exercício 5-c), o arranjo esteja totalmente ordenado.

Questão 7)

$f(n) = n^3/1000 - 100n^2 - 100n + 3$
$g(n) = n^3$ (limite assintotico restrito proposto)
$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{1}{1000} - \frac{100}{n} - \frac{100}{n^2} + \frac{3}{n^3} = \frac{1}{1000}$

Como $0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$, $f(n)$ é $\Theta(n^3)$.

Questão 8)

$c_1 n * \log(n) \leq \log 1 + \log 2 + \log 3 + \ldots + \log n \leq c_2 n * \log(n)$

Analizando o lado direito, claramente
$\log 1 + \log 2 + \log 3 + \ldots + \log n < \log n + \log n + \log n + \ldots + \log n = n * \log n$
ou seja, a inequação da direita é válida para $c_2 = 1$ para $n \geq 1$

O lado esquerdo
$\log 1 + \log 2 + \log 3 + \ldots + \log n \geq \log(n/2) + \log(n/2 + 1) + \ldots + \log n$ (ou seja, o loop todo tem o resultado das adições maior ou igual que apenas a segunda metade do loop)
$\log(n/2) + \log(n/2 + 1) + \ldots + \log n \geq \log(n/2) + \log(n/2) + \ldots + \log(n/2) =$
$= n/2 * \log(n/2) = n/2 * \log(n) - n/2 * \log 2$ que por sua vez também tem limite assintótico restrito $\Theta(n * \log n)$, e a propriedade da transitividade de limites assintóticos mostra que f(n) tem o limite assintótico inferior restrito igual a $n * \log n$.

Uma maneira alternativa é usando a fórmula de Stirling

$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

$\log 1 + \log 2 + \log 3 + \ldots + \log n = \log(1 * 2 * 3 * \ldots * n) = \log(n!)$
$\log(n!) \sim \log(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n) = \log(\sqrt{2\pi n}) + n * \log n - n * \log e$
o qual pela regra da soma de limites assintóticos é $\Theta(n * \log n)$

Questão 9)

$c_1 \max(f(n), g(n)) \leq f(n) + g(n) \leq c_2 \max(f(n), g(n))$
isso é equivalente a dizer
$c_1 \max(f(n), g(n)) \leq \max(f(n), g(n)) + \min(f(n), g(n)) \leq c_2 \max(f(n), g(n))$
dividindo todos os termos por $\max(f(n), g(n))$:
$c_1 \leq 1 + \frac{\min(f(n), g(n))}{\max(f(n), g(n))} \leq c_2$
Como estamos considerando o limite, então se $\min(f(n), g(n))$ tiver um limite assintotico de ordem menor, então a razão $\frac{\min(f(n), g(n))}{\max(f(n), g(n))}$ irá tender a zero, sendo desprezível. Caso tenha a mesma ordem, de $\max(f(n), g(n))$, a razão tenderá a uma constante, de modo que a equação no limite independentemente da ordem fique:

$c_1 \leq 1 + \mathrm{cte} \leq c_2$, de modo que existe c_1 e c_2 para $n \geq k$ satisfazendo o limite assintótico restrito.

Questão 10)

$c_1 n^b \leq (n+a)^b \leq c_2 n^b$
Usando a expansão do binômio de newton:
$c_1 n^b \leq n^b + \binom{b}{1} n^b a + ... + a^b \leq c_2 n^b$
Dividindo todos os termos por n^b temos:

$c_1 \leq 1 + \binom{b}{1} \frac{a}{n} + ... + \frac{a^b}{n^b} \leq c_2$
Para um valor suficientemente grande de n todos os valores da expansão com exceção do primeiro vão tender a zero, independente do valor de a, de modo que o limite assintótico igual a n^b independe do a.

Questão 11)

Sim, pois $2^{n+1} = 2 * 2^n$, e como estudado, constantes multiplicativas não interferem no limite assintótico.

Questão 12)

Não, pois segundo a definição, deveria ser verdadeira a seguinte condição:
$2^{2n} \leq c2^n$
Dividindo ambos termos por 2^n
$2^n \leq c$
E não existe nenhum valor de c que seja maior ou igual a 2^n para todo $n \geq k$, $k \geq 1$

Questão 13)

limite assintótico inferior:
existem constantes positivas c, n_0, m_0
tais que $0 \leq c\,g(n, m) \leq f(n, m)$
para todo $n \geq n_0$ e $m \geq m_0$
limite assintótico restrito:
existem constantes positivas c_1, c_2, n_0, m_0
tais que $0 \leq c_1 g(n, m) \leq f(n, m) \leq c_2 g(n, m)$
para todo $n \geq n_0$ e $m \geq m_0$

Questão 14)

Queremos provar a validade ou não de $(\log n)! > c * n^b$ (a hipótese inversa em que a função não é polinomialmente limitada) para todo $n \geq k$ para algum $k \geq 1$ (nota-se que o fatorial é sempre positivo).

Achando o logaritmo de ambos os lados da inequação temos:

$\log[(\log n)!] > \log(c * n^b)$
$(\log n)! = 1 * 2 * 3 * ... * \log n$
$\log[(\log n)!] = \log(1 * 2 * 3 * ... * \log n) = \log 1 + \log 2 + \log 3 + ... + \log\log n$
$\log(c * n^b) = b * \log(c * n) \sim c' * \log(n)$

$\log 1 + \log 2 + \log 3 + ... + \log\log n > c * \log(n)$ (reescrevendo a hipótese original)
e
$\log 1 + \log 2 + \log 3 + ... + \log\log n \geq \log(\frac{\log n}{2}) + \log(\frac{\log n}{2} + 1) + ... + \log\log n \geq \frac{\log n}{2} * \log(\frac{\log n}{2})$
(ver exercício 8, primeiro método de resolução)
Ou seja, achamos uma expressão menor ou igual a $\log[(\log n)!]$, de modo que se essa for maior que $c * \log \mathrm{n}$, então aquela tambem é.

$\frac{\log n}{2} * \log(\frac{\log n}{2}) > c * \log n$ (dividindo ambos os lados por $\log n$)
$\frac{1}{2} * \log(\frac{\log n}{2}) > c$

O que obviamente éverdadeiro pois a função log n tende ao infinito quando n tende ao infinito.

A resposta então é que $(\log n)!$ não é polinomialmente limitada.

Questão 15)

Queremos provar a validade ou não de $(\log\log n)! \leq c * n^b$ (a hipótese de que a função é polinomialmente limitada) para todo $n \geq k$ para algum $k \geq 1$ (nota-se que o fatorial é sempre positivo).

Achando o logaritmo de ambos os lados da inequação temos:
$\log[(\log\log n)!] \leq \log(c * n^b)$
$(\log\log n)! = 1 * 2 * 3 * ... * \log\log n$
$\log[(\log\log n)!] = \log(1 * 2 * 3 * ... * \log\log n) = \log 1 + \log 2 + \log 3 + ... + \log\log\log n$
$\log 1 + \log 2 + \log 3 + ... + \log\log\log n \leq \log\log n * \log\log\log n = f'(n)$ (a soma de elementos de 1 até o último elemento é menor que o último elemento multiplicado pelo número de elementos)
Como achamos uma expressão maior ou igual a $\log[(\log\log n)!]$, se essa for menor que $c * \log \mathrm{n}$, então aquela tambem é.
$\log(c * n^b) = b * \log(c * n) \sim c' * \log(n)$

A condição suficiente para o limite assintótico superior é:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$$

O que nesse caso produz o mesmo resultado que $\lim_{n \to \infty} \frac{\log f(n)}{\log g(n)}$ e $\lim_{n \to \infty} \frac{f'(n)}{\log g(n)}$.

Então, com ajuda da regra de l'Hôpital:

$$\lim_{n\to\infty}\frac{f'(n)}{\log g(n)} = \lim_{n\to\infty}\frac{\log\log n * \log\log\log n}{\log n} = \text{(derivando)}$$
$$\lim_{n\to\infty}\frac{\frac{\log\log\log n+1}{n*\log n}}{\frac{1}{n}} = \lim_{n\to\infty}\frac{\log\log\log n + 1}{\log n} = \text{(derivando)}$$
$$\lim_{n\to\infty}\frac{n}{n*\log n * \log\log n} = \lim_{n\to\infty}\frac{1}{\log n * \log\log n} = 0$$

Como o limite é limitado, a hipótese é verdadeira.

A resposta então é que $(\log\log n)!$ é polinomialmente limitada.

Questão 16)

A comparação entre dois pode ser feita calculando os limites das razões (ou usando técnicas exemplificadas nas questões anteriores). A solução fica:

$$1 = n^{\frac{1}{\lg n}} < \lg * \lg n < \lg * n < \lg\lg * n < 2^{\lg * n} < \ln\ln n < \sqrt{\lg n} < \ln n < 2^{\sqrt{2 * \lg n}} < \sqrt{2}^{\lg n} < (\lg n)^2 < 2^{\lg n} = n = 2n < 4^{\lg n} < n * \lg n = \lg(n!) < n^2 < n^3 < (\lg n)! < (\lg n)^{\lg n} = n^{\lg\lg n} < \left(\frac{3}{2}\right)^n < e^n < n * 2^n < 2^{2n+1} < n! < (n+1)! < 2^{2^n}$$

obs: lg* é o chamado logaritmo iterado.

Questão 17)

a)
Sempre verdadeira
$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{f(n)}{f(n)^2} = \lim_{n\to\infty}\frac{1}{f(n)} < \infty$$ (se $f(n)$ não tender a zero)
Na prática no entanto não existe funções descrevendo algoritmos cujo número de operações tendam a zero quando n aumenta, então essencialmente é sempre verdadeira.

b)
Sempre verdadeira
É uma das propriedades dos limites assintóticos (regra da soma), e provada na questão 9.

c)
Ás vezes falso, às vezes verdadeiro
Ela só é verdadeira quando $O(f(n))$ for da menor ordem possível
Exemplo verdadeiro:
$f(n) = 2n + 3; O(f(n)) = n$
$3n + 3 = \Theta(n) = \Theta(f(n))$
Exemplo falso:
$f(n) = 2n + 3; O(f(n)) = n^2$

$n^2 + 2n + 3 = \Theta(n^2)! = \Theta(f(n))$

d)
Sempre falsa
A primeira condição exige:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$$

enquanto a segunda exige a condição contrastante:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

e)
Sempre falso

A condição suficiente para ser $O(f(n))$ é:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

Então para não ser é:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Para ambas serem verdadeiras, seria preciso também que:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$$

E não existem funções reais $f(n)$ e $g(n)$ que satisfaçam essas duas condições.

Questão 18)

Verdadeiro.

$f(n) \leq c * g(n)$ ou $g(n) \leq c * f(n)$

Suponhamos que a primeira inequação é falsa
$f(n) > c * g(n)$
$g(n) < \frac{1}{c} f(n)$
$g(n) \leq c' * f(n)$
E a segunda inequação se torna verdadeira (e vice-e-versa).