***Entregar todas as respostas em um único arquivo pdf.***

1) (1,5 ponto) Verifique se as afirmações abaixo são verdadeiras ou falsas, justifique as falsas.

   a) $\textit{NP-Hard} \cap \textit{NP-Completo} = \varnothing$.

   b) Se o problema da fatoração de inteiros grandes for resolvido em tempo polinomial então $P = NP$.

   c) Se $P = NP$ então qualquer problema pode ser resolvido em tempo polinomial.

   d) Considerando dois problemas $A$ e $B$, se $A \leq_p B$ e $A \in P$ então $B \in P$.

   e) Considerando dois problemas $A$ e $B$, se $A \leq_p B$ e $B \in P$ então $A \in P$.

   f) $\textit{NP-Hard} \subseteq NP$.

2) (1,5 ponto) Resolva a seguinte relação de recorrência:

$$T(n) = T(n/2) + n^2$$
$$T(1) = 1$$

3) (2,0 pontos) Complete (em linguagem C) a função de inserção em um *heap* binário máximo implementando a função **heapfyLeaf** que reestabelece a propriedade de *heap* quando um elemento é inserido. Qual a complexidade de tempo e espaço da função insertHeap considerando sua implementação?

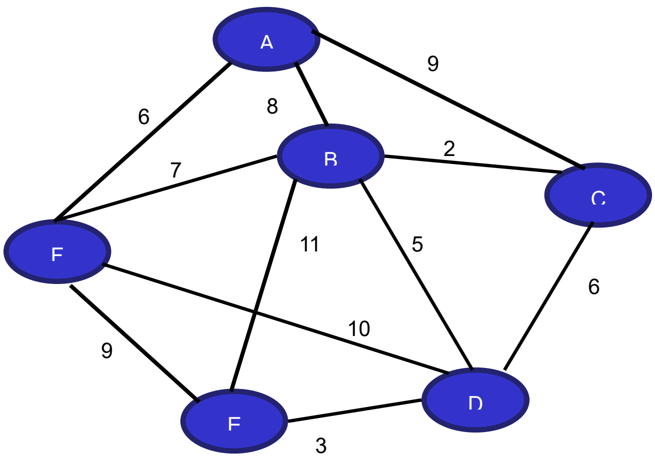
```
int parent(int i)
{
  return ((i - 1) / 2);
}

/* Sendo a o vetor que implementa o heap, n o tamanho do heap, l o tamanho do vetor e e o elemento a ser inserido */

void insertHeap(int *a, int n, int l, int e)
{
        if (n < l)
        {
                a[n] = e;
                heapfyLeaf(a, n);

        }
        else
        {
                printf("insertHeap: Heap overflow!");
                exit(1);
        }
}
```

4) (2,0 pontos) Mostre cada passo da execução do algoritmo de *Kruskal* para encontrar a árvore geradora mínima do grafo abaixo. Mostre o algoritmo e a complexidade de tempo de cada *loop* e função auxiliar. Informe a complexidade de tempo do algoritmo.




5) (1,0 ponto) Mostre a redução do problema *2-CNF-SAT* ao *3-CNF-SAT*, usando como exemplo a instância de *2-CNF-SAT* abaixo. Essa redução demostra que *2-CNF-SAT* pertence à classe *NP-Hard*? Explique.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$$

6) (2,0 pontos) Um **conjunto independente de vértices** de um grafo simples $G = (V, A)$ é um subconjunto de vértices $V'$ de $G$ tal que não existem dois vértices adjacentes contidos em $V'$. Em outras palavras, se $u \in V'$ e $v \in V'$ então $(u, v) \notin A$. Prove que determinar se em um grafo existe um conjunto independente de vértices de tamanho $k$ é um problema que pertence à classe *NP*.