(a) (T) Every context-free language is recursive.
解析：上下文无关语言包含于递归语言，所以正确，反之不成立，另外正则语言包含于上下文无关语言。所以说正则的也是递归的也是正确的。

(b) (T) Language $\{a^m b^n c^l \mid m, n, l \in \mathbb{N}, m+n > 3l\}$ is context free.
解析：因为证明语言是上下文无关的，可以根据 引理 3.4.2 ，构造一个下推自动机来接受它，或者，利用 定理 3.5.1，上下文无关语言对于并封闭，把原来的语言分解为几个语言的并，证明这几个语言是上下文无关，并回来也是 context free。这条题目可以用第一个方法。遇到 a 和 b 都入栈，当然不按顺序就要拒绝了，然后遇到一个 c 就出栈三个，如果字符串读完之后栈不空，那么就接受。于是得证。

(c) (T) Every language in NPis recursive.
解析：因为递归语言包含了 NP 语言，所以正确。

(d) (F) All languages on an alphabet are recursively enumerable.
解析：存在非递归可枚举的语言。例如课本上 P164 的 H1 的补，它不是递归可枚举的。因此，不能说所有字母表上的语言都是递归可枚举的。

(e) (F) There's a language L such that L is undecidable, yet L and its complement are both semi-decided by the some Turing machine.
解析：定理 5.7.1 语言是递归的当且仅当它和它的补都是递归可枚举的。所以如果 L 和其补都可能被图灵机半判定，等价于存在 TM 判定 L。

(f) (T) There's a function φ such that φ can be computed by some Turing machines, yet φ is not a primitive recursive function.
解析：原始递归函数真包含于 μ 递归的，μ 递归的函数就是递归的函数，可以被图灵机计算。这句话的意思是，存在非原始递归的可计算函数，因此是正确的。

(g) (F) Let $L_1, L_2 \subseteq \Sigma^*$ be languages, recursive function t is a reduction from L1 to L2, if L1 is decidable, then so is L2.
解析：如果 L1 通过一个递归函数归约到 L2，那么如果 L2 是递归的则 L1 也是递归的。归约和判断的方向是相反的。另外可以有逆否命题，如果 L1 不是递归的那么 L2 也不是递归的

(h) (F) A language L is recursive if and only if it is Turing-enumerable.
解析：定理 5.7.2 ，语言是递归可枚举的当且仅当它是 Turing 可枚举的。所以这题错误。另外，定理 5.7.3 语言是递归的当且仅当它是以字典序 Turing 可枚举的。

(i) (F) Suppose A, B are two languages and there is a polynomial-time reductions from A to B. If A is NP-complete, then B is NP-complete.
解析：应该是反过来说。如果 A 多项式时间归约到 B，那么如果 B 是 NPC，那么 A 也是 NPC。把 NPC 换成 P 或 NP 同理。另外根据定义 7.1.2 ，可以说若 B 是 NPC 的，那么 A 是

NP 的。

(j) (T) Every language in NP-complete can be reducible to the 3-SAT problem in polynomial time.
解析：定理 7.2.3，三元可满足性是 NP 完全的。所以，所有 NP 问题都可以多项式时间归约到 NPC，而三元可满足性是 NP 完全的，所以所有 NPC 都可归约到 3-SAT。

(a) Decide whether the following language is regular or not and provide a formal proof for your answer.

$$L = \{a^m b^n \mid m, n \in \mathbb{N}, (m-n) \mod 3 \neq 0\}$$

解析：首先这不是正则的。
然后，证明非正则大概有两种办法，第一是用定理 2.4.1，也就是正则版本的泵定理，证明其不是变成 xy^iz，第二就是用补运算的封闭性，证明其补不是正则的，然后其本身也不是正则。
这条题目用泵定理。把 $w=a^mb^n \in L$ 重写成 xyz，使得 $|xy| \leqslant n$ 且 $y \neq e$，那么让 $y=a^i$，但是 $xz=a^{m-i}b^n$ 不一定 $\in L$，让 $y=b^i$ 同理，所以不是正则的。

(b) Let Σ be an alphabet and let $L_1, L_2 \subseteq \Sigma^*$ be languages so that L1 is not regular but L2 is regular. Assume L1 ∩ L2 is finite. Prove that L1 并 L2 is not regular.

(b) Assume $L_1 \cap L_2$ is finite. Since every finite set is regular, $L_1 \cap L_2$ is regular. Observe that

$$L_1 = ((L_1 \cup L_2) - L_2) \cup (L_1 \cap L_2).$$

If $L_1 \cup L_2$ were regular, since the regular languages are closed under the operations union, intersection and complement and since L_2 and $L_1 \cap L_2$ are regular, L_1 would be regular, a contradiction. Therefore, $L_1 \cup L_2$ is not regular.

3. **(20%) PDA and Context-free languages:**

(a) Give a context-free grammar for the language

$$L_3 = \{xy \mid x, y \in \{a, b\}^*, |x| = |y| \text{ and } x \text{ and } y^R \text{ differ in one positions }\}.$$

For example, $abbbbaba, abbbbbbb \in L_3$, but $aababb \notin L_3$.

(b) Design a PDA $M = (K, \Sigma, \Gamma, \Delta, s, F)$ accepting the language L_3.

Solution: (a) We can construct the context-free grammar $G = (V, \Sigma, R, S)$ for language L_3, where

$V = \{a, b, S, A, B\}; \Sigma = \{a, b\}$; and

$$R = \{S \to aSa, S \to bSb, S \to aAb, A \to aAa, A \to bAb, A \to e,$$

$$S \to bBa, B \to aBa, B \to bBb, B \to e\}$$

(b) The PDA $M = (K, \Sigma, \Gamma, \Delta, s, F)$ is defined below:

$K = \{p, q\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, S, A, B\}$

$s = p$

$F = \{q\}$

(q, σ, β)	(p, γ)
(p, e, e)	(q, S)
(q, e, S)	(q, aSa)
(q, e, S)	(q, bSb)
(q, e, S)	(q, aAb)
(q, e, A)	(q, aAa)
(q, e, A)	(q, bAb)
(q, e, A)	(q, e)
(q, e, S)	(q, bBa)
(q, e, B)	(q, aBa)
(q, e, B)	(q, bBb)
(q, e, B)	(q, e)
(q, a, a)	(q, e)
(q, b, b)	(q, e)

解析：按题目要求写出响应的文法并不难，只需要记着格式，就是 G={V， Σ，R，S}，然后分别说明 V 里面有什么字母，Σ里面有什么非终结符，R 里面是什么规则，S 就是起始符了。

然后，根据上下文无关语法构造相应下推自动机的方法，可以参照引理 3.4.1，注意格式。M={K， Σ，Γ，Δ，s，F}，然后 K 是机器会处于的状态，Σ是终结符，Γ是字母，s 是其实状态，F 是终结状态，都一一说明，并列出表格。

4. **(10%) Numerical Functions:**
Let $P(x, y)$ be primitive recursive predicates. Prove the following predicate

$$\forall y_{\le u} P(x, y)$$

is also primitive recursive.

solution:

$$\forall y_{\le u} P(x, y) \Leftrightarrow \Pi_{y=0}^{u} P(x, y) \neq 0$$

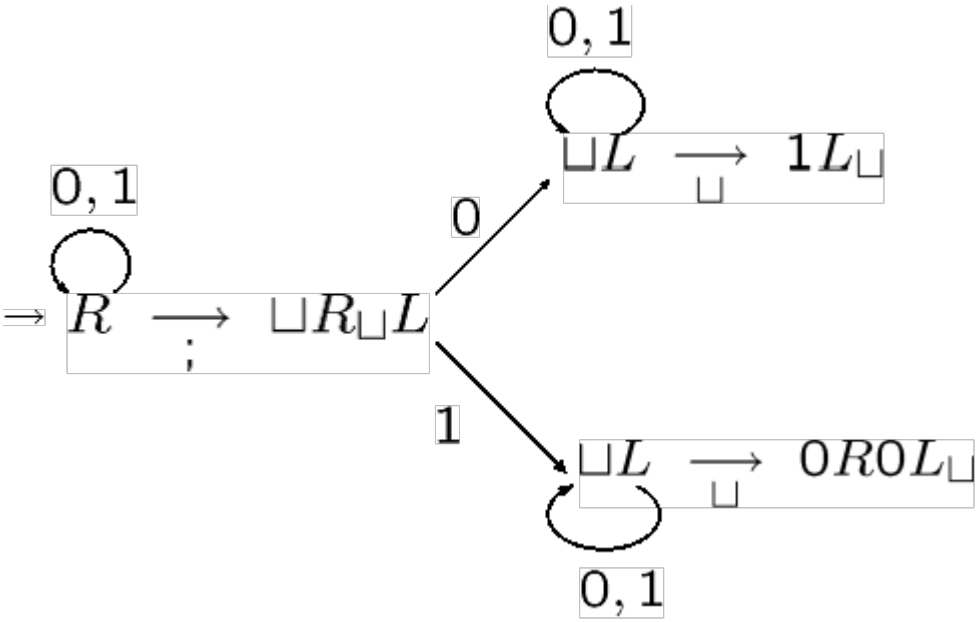
解析：P(x,y)是原始递归谓词。显然，任意的 y≤u，P(x,y)都成立，等价于 $\Pi_{y=0}^{u} P(x, y) \neq 0$，因为原始递归谓词的合取（或析取）都是原始递归谓词，因此得证。

5. **(10%) Turing Machines**

Design a Turing machine for computing the following function.

$$f(x,y) = \begin{cases} 2x+1, & \text{if } y \text{ is even} \\ 4x, & \text{if } y \text{ is odd} \end{cases}$$

where x and y are represented by binary strings respectively and separated with the symbol ";", i.e. the initial configuration in form of $\triangleright \underline{\sqcup} x; y$.



解析：因为输入的字符串是 空格 x;y，都是二进制的

因此，循环向右，直到遇到分号;

然后把分号变成空格，向右寻找空格，找到之后向左移一格

如果是 0，那么表示 y 是偶数，然后把当前位置变空格并左移，改空格并左移，直到遇到空格，这是 x 要变成 2x+1，其实就是左移一位加一，这里就是把原来分号的地方（现在是空格）变成 1，然后左移直到找到空格

如果是 1，那么表示 y 是奇数，同样把 y 消灭了，然后 4x 就是在最后加两个 0，然后左移直到找到空格。

6. **(10%) Decidability and Undecidability**

Show that the following language

$$\{\text{“}M\text{”}\ \text{“}w\text{”} \mid M \text{ is a TM and } M \text{ halts on } w\}$$

is recursively enumerable. An informal description suffices.

Solution: The universal Turing machine U can semidecides the language

$$\{\text{“}M\text{”}\ \text{“}w\text{”} \mid M \text{ is a TM and } M \text{ halts on } w\}$$

解析：suffice 的意思是足够，最后一句就是非正式的描述都 ok 了。

证明一个语言是递归可枚举的，只要找到一个半判定这个语言的图灵机就可以了。这条题目就可以用通用图灵机 UTM。

稍微说说通用图灵机，简称通灵机。举个例子，大家知道 Java，Java 的代码编译成字节码，然后在不同系统上的 Java 虚拟机 JVM 上运行，而 JVM 本身就是个程序，我们的电脑可以根据 JVM 这个程序处理给定输入给出结果。同样的，我们把一个图灵机现在变成一个程序，通用图灵机就是我们的电脑，它接受输入就是图灵机的程序及给这个程序的输入，然后通用图灵机就告诉我们结果。可以这么说，“M”是虚拟机，通用图灵机就是虚拟机的宿主机。

7. **(10%)$\mathcal{P}$ and $\mathcal{NP}$ Problems**

Given n natural numbers $x_1, x_2, \cdots, x_n$ to test whether there exist distinct $i_1, i_2, \cdots, i_k$ such that $x_{i_1} + x_{i_2} + \cdots + x_{i_k} = (x_1 + x_2 + \cdots + x_n)/2$, and $x_{i_1} + x_{i_2} + \cdots + x_{i_k}$ is not a prime number. Design a $\mathcal{NP}$ algorithm for it, and estimate its time complexity.

Solution: First compute

$$H = \frac{1}{2}\sum_{i=1}^{n} x_i$$

in $\mathcal{O}(n)$ steps.

Then guess $i_1, i_2, \cdots, i_k$ for some $k \leq n$, this takes $\mathcal{O}(n)$ steps, and compute $S = x_{i_1} + x_{i_2} + \cdots + x_{i_k}$ then check $S = H$.

If so, guess a factor x of S and then divide S by x and verify that the remainder is 0. If all the tests succeed print “yes” and halt else simply halt.

observe that if the answer is “yes”, our algorithm can print “yes” and if the answer is “no”, our algorithm cannot print “yes”.