1. EXAMPLES: Give examples of each of a, b and c below. (Two points each.)
   You need not justify your answers, but you should precisely define the
   languages you are using as examples, so that it is clear exactly which strings are
   in these languages and which are not.

   For instance, if you are using a set S as an example, then write something like
   \[ S = \{ (M,x) \mid M \text{ is a TM and } x \text{ is a binary string and } M \text{ accepts exactly those}
   \text{ strings } x \text{ which have 0100 as a substring } \}. \]

   You can say it with words as well but you need to be clear on what you mean.

   a. A decidable language L and an undecidable language K where L - K is
      undecidable. Answer: Let L the set of all strings in the alphabet of the halting
      problem and let K = the halting problem H.
      Then L - K is the complement of the halting problem which is undecidable since
      H is.

   b. Give an example of some evidence that Church/Turing’s thesis is true.

      Answer: Here lots of examples suffice. Saying that TM’s are robust in the sense
      that any algorithmic extension of the standard TM has been shown to be
      computable by the standard TM is fine, as is saying any program in ay
      programming languages implemented in the past 70 years has been shown to be
      programmable by a TM. This works too.

   c. A subset of the accepting problem \( A_{TM} \) which is infinite and decidable.

      Answer: Let T be TM which accepts every binary string x. Let S = \{ (T,x) \mid x
      is a binary string \}. \]

      Then S is infinite and as T accepts every binary string S \( \subseteq A_{TM} \), and S is
      clearly decidable.
2. Say whether each of i. - iv. is True or False (1 points each, no partial credit)

Answers: The answers are F T T F

i. If Church/Turing’s thesis is true then any language which is recognized by a C program TM can be decided by a TM.

ii. There are infinitely many different undecidable problems.

iii. If A is recognizable and B is decidable then C = (A ∪ B) is recognizable.

iv. A subset of a recognizable set is always recognizable.

3. One of the following sets B or C is recognizable and one is not. (3 points)
   Pick which one is recognizable and briefly give an algorithm (or describe a TM) to recognize it.

B = \{ <M> | M accepts no more than 5 different inputs \}
C = \{ <M> | M accepts greater than 5 different inputs \}

Answer: The language C is recognizable. To recognize it, given M we need to give an algorithm which accepts if and only if M accepts greater than 5 different inputs.

To do this, given M we run M on all of its input interweaving the computations and running them for more and more steps. As we do this we keep a counter which keeps track of the number of different inputs x where M(x) accepts. If this counter ever gets to a number bigger than 5 we halt and accept.

If not the “interweaved simulation of M” runs on forever and never accepts (or even halts).

4. (2 points) Which one of the following 4 statements is NOT true about a universal Turing machine (UTM).

a. The set of inputs for which the UTM halts is recognizable.

b. You could actually write the program of such a machine.

c. A UTM can be used to decide an unsolvable problem.

d. The UTM can simulate the computation of any Turing machine on any input.

Answer: c is not correct as a UTM is a specific TM and an unsolvable problem is one which cannot be accepted by any TM.
5. (Somewhat longer answer, 5 points) Pick ONE of the problems $P_1$ or $P_2$ below to answer. Do not answer the other one.

$P_1$:

Prove that the language 
\[ \{(M,w)| M \text{ is a Turing machine and } w \text{ an input to } M \text{ and } (M,w) \text{ only moves to the right during its computation}\} \] is decidable.

Answer: VERY briefly - to decide if $(M,w)$ is in the language $J =$ 
\[ \{(M,w)| M \text{ is a Turing machine and } w \text{ an input to } M \text{ and } (M,w) \text{ only moves to the right during its computation}\} \]

We run the computation of $M(w)$ for $|w|+|Q|+1$ steps. ($|Q|$ is the number of states that $M$ has).

If the computation ever moves left during this computation then we halt and reject $M(w)$.

If the computation ever halts during the computation and never moves left (that is it only moves to the right) then we accept $M(w)$.

If the computation never halts during the $|w|+|Q|+1$ steps of the computation then we halt our algorithm and accepts.

What’s left to say is why this procedure is correct and decides the language $J$ above.

$P_2$:

Here is the TM $M$ which was discussed in class. It decides the collection of all binary string which contains an odd number of 1’s.

$M$ has 4 states $Q = \{q_0, q_1, q_a, q_r\}$, two input symbols 0 and 1, three tape symbols 0,1,and B, and M’s program consists of the following 6 five-tuples.

\[
(q_0, 0, q_0, 0, R) \quad (q_0, 1, q_1, 1, R) \quad (q_1, 0, q_1, 0, R) \\
(q_1, 1, q_0, 1, R) \quad (q_0, B, q_r, 0, R) \quad (q_1, B, q_a, 0, R).
\]

You job is to extend $M$ to a TM $N$ that it decides the language $L$ where 
\[ L = \{ x|x \text{ is a binary string containing an odd number of 1’s and having two or more 0’s in it } \}. \] So for example the string 100101 is in $L$ but 110111 is not in $L$.

You should either write the program TM for $N$ either directly as above, or using the graph diagram for the program as in Sipser’s textbook. Also, state what $N$’s set of states and alphabet is.