At the top of your homework paper write your name, the class number (CS 332) and also the homework assignment number (HW 4), and write the number of the problem at the top of your problem solution for each problem.

Reading: Chapter 5: Section 5.1, only pages 215-218, and section 5.3, pages 234-238. Chapter 7: Pages 275-290.

PROBLEMS:

1. i. Give an example of two undecidable languages whose union is decidable.
   
   Hint: Consider using HALT or $A_{TM}$ as one of the sets.
   
   Answer: The Halting problem and its complement (both undecidable) work here. Their union is the set of all string of the form ¡ ¡M¿,w¿ which is easily decidable.

   ii. Give an example of two undecidable languages whose intersection is infinite and decidable.
   
   Answer: Consider HALT, now find an infinite decidable language L1 which is disjoint from Halt. So, for example L1 could be {!}$^*$. Note that the symbol ! is not even in HALT’s alphabet.

   Then $L1 \cup$ Halt and $L1 \cup$ (the complement of HALT) are both undecidable and their intersection is $L1$ which is infinite and decidable.

   Why? HALT and the (Complement of HALT) are both undecidable and disjoint, and because of disjointness $L1 \cup$ Halt and $L2 \cup$ Halt are both also undecidable.

2. We say a function $f:N \rightarrow N$ is total computable if there is TM M which start with an natural number n on its tape and computes $f(n)$ on its tape (and halts).

   For example the function $f(n) = n+1$ is total computable.

   If we allow the TM M to loop on some inputs the function is called partial computable.

   Explain why the range of any total computable function is a recognizable language.

   Answer: Let f be a total computable function.

   Then range(f) = \{ y | for some x ∈ N, f(x)=y \}. We need to show that range(f) is acceptable.

   There are two (somewhat different) way to do this.

   Method 1: We proved in class than any enumerable set is acceptable.

   It is straightforward to enumerate range(f). To do this let M be the TM which computes f.

   For i = 1,2,3,.....

   run M(i) until it halts with f(i) on its tape and then add f(i) to your enumeration.
Method 2: Here we directly show that range(f) is acceptable by constructing an algorithm which accept exactly the natural numbers in range(f).

**ALGORITHM:** For an input $w \in \mathbb{N}$,

(i) Let $i=1$.

(ii) Run $M(i)$ until it halts and computes $f(i)$. If $f(i) = w$ then halt and accept $w$. Otherwise, $M(i)$ must halt computing an output not equal to $w$. Then let $i=i+1$ and repeat step (ii).

This ALGORITHM accepts exactly range(f). (Note that if $w$ is not in the range of $f$ then our algorithm loops in input $w$.)

3. Recall that the language $HALT_{TM} = \{(< M >, x) | M \text{ is a TM and } x \text{ is an input to } M \text{ and } M(x) \text{ halts } \}$ $HALT_{TM}$ was shown in class to be undecidable. (Also on page 216, Theorem 5.1 in the Sipser book.)

i. Is $HALT_{TM}$ a finite language or an infinite language?

ii. Let $S \subseteq HALT_{TM}$. What do the elements of $S$ look like? 
   e.g. “An element of $S$ is a pair $\text{such that}$”.

iii. Define a subset $S$ of $HALT_{TM}$ which is infinite and decidable.
   Note: You need to say/define here which elements of $HALT$ are in your set $S$, not just that such an $S$ exists.

4. Prove that $L = \{ < M > | M \text{ accepts the string 10} \}$ is undecidable
   Answer: (Note: This answer is incomplete, but more details will be provided soon.)
   We will prove $L$ is undecidable by reducing the undecidable language $A_{TM}$ to $L$.
   Assume we have a decider $T$ which decides $L$. We present a reduction $A$ which uses $T$ and an input $< < M >, x >$ to the reduction to decide whether $< < M >, x >$ is in $A_{TM}$.

5. Prove that $S = \{ < M > | M \text{ never enters the state } q_4 \text{ on any input string } \}$ is undecidable.
   Answer: Pf: Mentioned in class, and in the textbook on page 217, Theorem 5.2, it is shown that $E_{TM}$ is undecidable.
   $E_{TM}$ is the problem of determining whether the set that a TM $M$ accepts is just the empty set (that is, it M accepts no strings at all.)
   To use $E_{TM}$ to show that $S$ is undecidable by reducing $E_{TM}$ to $S$. To do this we assume $S$ is decidable and using this fact explain hot to decide the set $E_{TM}$.
   Let $M$ be a TM. We change $M$ to a machine $M^*$ in the following 2 steps, slightly changing $M$’s program both times.
   1. First, if there is a state $q_4$ in $M$’s set of states we replace all the $q_4$’s in $M$’s program with some new state, call it p. (So really M works just as before using p but now there is no state $q_4$ anywhere in it’s program.)
2. Now we go through M’s program again, this time bringing back into use the state \( q_4 \) in a special way.

Namely any time there is a program step which results in M going into its accept state \( q_a \) we add an extra step in M’s program which makes it first go into state \( q_4 \) (instead of \( q_a \) and then we add one extra line in its program which causes it to go into \( q_a \) whenever it is in state \( q_4 \).

This 2 step change to M results in a machine we call M*. Note that M8 has the funny property that on any input \( w \), \( M^*(w) \) accepts if and only if it goes into state \( q_4 \) during its computation. (In fact, right before it goes into state \( q_a \).

So now we can use this to decide \( E_{TM} \). The algorithm to do this is:

Take any TM M. we want to know if M is in \( E_{TM} \). First change M to M*. Then use the decidability of S to tell us if M* is in S and we accept M if and only if M* is in S. Note that M is in \( E_{TM} \) if and if the language accepted by M is the empty set if and only if

Reduce the accepting problem \( A_{TM} \) to S. That is we need to find a computable function f where

\[
<M, w> \text{ is in } A \text{ if and only if } f(<M, w>) = <M^*> \text{ is in } S.
\]

Here is how we define f in an input \(<M, w>\).

We use M to define the new TM \( M^* \) as follows.

0. We start with \( M^* = M \). If \( 5 \) is not in M’s tape alphabet add it to the tape alphabet.

1. Add a new element to \( M^* \)’s tape alphabet, call it +. Everywhere that 5 occurs in M’s program, replace the 5 with a +.

2. Also, change \( M^* \)’s program so that given any input \( x \) to \( M^* \), it first checks if 5 is in the input string \( x \) and replaces any 5’s with +’s.

So now \( M^* \) on any input \( x \) computes just like \( M(x) \) except it uses +’s where \( M \) used 5’s.

3. Finally change \( M^* \)’s program so that just before is ever goes into the state accept, it first writes 5 on its tape and then accepts at the next step.

Now define \( f(<M, w>) = <M^*, w> \). It should be clear that \( M(w) \) accepts if and only if \( M^*(w) \) writes a 5 and then accepts if and only if \( M^*(w) \) writes a 5 on its tape during its computation.

So \( f \) is the reduction which is our goal.