Massachusetts Institute of Technology
Department of Aeronautics and Astronautics
Cambridge, MA 02139

Unified Engineering
Spring 2005

Problem Set #9

Due Date: Tuesday, April 12, 2005 at 5pm

	Time Spent (minutes)
S7	
S8	
S9	
S10	
C8	
C9	
C10	
Study Time	

Name: ________________

The problems in this problem set cover lectures C8, C9, and C10.

Download and unzip the following file from the C&P class webpage:

`C&P_PSet3_Files.zip`

Some answers (C6c, C7a, C7b) will need to have their answers zipped and uploaded to:

https://spacestation.mit.edu/unified/

Three *modified* files are expected:

`merge_sort.adb,`
`doubly_linked_list.ads AND doubly_linked_list.adb`

Please zip/compress/store these three .adb files. Name the file:

"C910_Lastname_Firstname.zip"

For this problem set, there are no implementation rules. There are no specific instructions as to whether the algorithms should be done recursively or iteratively nor do we say anything about what parameters they should take.

Problem C8. Algorithm Complexity

Compute the complexity of the code shown below by hand, using:

a. Iteration
b. Simplified Master Method (last slide in lecture 9)

```
procedure My_New_Sort(

      A : in out my_array; left: in integer; right in integer

      )is

first_split : integer;

second_split : integer;

begin

      if (left < right) then

            first_split := (left + right) / 3 ;

            second_split := (first_split + right)/2;

            My_New_Sort(A, left, first_split);

            My_New_Sort(A, first_split+1, second_split);

            My_New_Sort(A, second_split+1, right);

            Merge(A, left, first_split, second_split, right);

      end if;

end My_New_Sort;
```

Problem C9. Sorting Arrays

With the following files:

- `merge_sort.ads`
- `merge_sort.adb`
- `merge_test.adb`

a. Write an algorithm (pseudo-code) for merging two sorted arrays A, B size *n* to create a merged array C of size *2n*.

For example:

A	1	3	5	7				
B	2	6	45	55				
C	1	2	3	5	6	7	45	55

b. Modify your algorithm (pseudo-code) in part a. to merge two consecutive segments of an array, that are specified by indices called *lb*, *mid* and *ub*, where *lb* stands for left bound, *mid* stands for the mid point, and *ub* stands for upper bound.

For example:

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
My_Array	1	4	3	5	23	2	45	22	3	43

Note that both array segments My_Array(1..2), and My_Array(3..4) are sorted. Note that in cases where the array elements cannot be divided evenly, an arbitrary choice should be made as to which point is considered the midpoint.

The call Merge(My_Array,1,2,4) returns the merged array as follows

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
My_Array	1	3	4	5	23	2	45	22	3	43

Note that the array segment My_Array(1..4) is now sorted.

c. Implement the algorithm from part b as the merge procedure in body of the merge sort package.

Turn in a hardcopy for part a and part b (your algorithm/pseudo-code), as well as a hardcopy code listing of `merge_sort.adb`. Submit an electronic copy of `merge_sort.adb` online.

Problem C10. Sorting Linked Lists

a. Write an algorithm (pseudo-code) for performing **insertion sort** on a **doubly linked list**.

b. In Problem C4, you created a package for creating and manipulating doubly linked lists. Modify your doubly_linked_list package specification to include a subprogram for insertion sorting a doubly linked list.

c. Implement the sorting subprogram/procedure in the doubly_linked_list package body.

Turn in a hardcopy for part a (your algorithm/pseudo-code), as well as a hardcopy code listing of `doubly_linked_list.adb and doubly_linked_list.ads`. Submit an electronic copy of both online.

Unified Engineering II **Spring 2005**

Problem S7 (Signals and Systems)

This problem shows why a radar system sends out a chirp, which has a broad range of frquencies in the signal, and *not* a short sinusoidal pulse, which is at a single frequency. To see why a sinusoidal pulse doesn't work well, let's try a radar signal

$$u(t) = \begin{cases} \sin(2\pi t), & -3 \le t \le 0 \\ 0, & \text{otherwise} \end{cases}$$

The matched filter for this pulse has impulse response

$$g(t) = u(-t) = \begin{cases} \sin(-2\pi t), & 0 \le t \le 3 \\ 0, & \text{otherwise} \end{cases}$$

The radar sends out a signal, $u(t)$, that reflects off the aircraft and returns to the radar system. The time it takes the signal to return is twice the distance to the aircraft, divided by the speed of light. The received signal is $u(t-T)$, where T is the round trip travel time of the signal. For the purposes of this problem, we can ignore the time delay, T, and just look at how the matched filter response to $u(t)$.

1. Find the convolution
$$y(t) = g(t) * u(t)$$
You will find it helpful to use the flip and slide method to set up the integral. The integral can be evaluated relatively easily in closed form, if you set up the integral properly.

2. Plot $y(t)$.

3. $y(t)$as plotted above is the signal that results when the round-trip time of the pulse is zero. When the delay time is greater, of course, the signal that results is $y(t-T)$, which is just $y(t)$ shifted right by T. What feature of $y(t-T)$ would you use to identify the time T?

4. Explain why it might be difficult to determine T from a returned radar pulse, especially if there is additional noise added to the signal.

5. The signal $y(t)$ as plotted in Part 2 is called the *ambiguity function*, because it helps determine how ambiguous the delay time T is in the presence of noise. Explain why the ambiguity function corresponding to the chirp signal of Problem S6 is better than the ambiguity function in this problem.

In practice, the chirp signals sent out by a radar pulse are longer than a few cycles of Problem S6 or this problem. Nevertheless, these problems show that in order to have low ambiguity, the radar signal needs to have a well-designed shape.

Problem S8 (Signals and Systems)

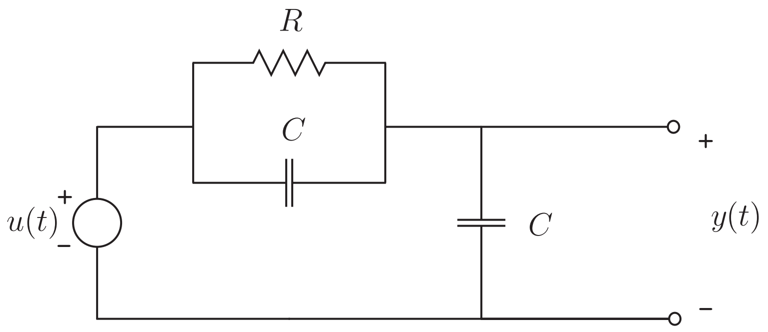
For each of the functions below, find the Laplace transform of the function, as well as the region of convergence. Do not use any outside reference (table of transforms or the Unified bible) to find the answer. You may use a table of Laplace transforms to *check* your work. However, show the derivation of the result, *i.e.*, work out the Laplace integral. Make sure that you include the region of convergence.

1. $g(t) = \begin{cases} te^{-at}, & t \geq 0 \\ 0, & t < 0 \end{cases}$
2. $g(t) = \begin{cases} t^2e^{-at}, & t \geq 0 \\ 0, & t < 0 \end{cases}$
3. $g(t) = \begin{cases} t^ne^{-at}, & t \geq 0 \\ 0, & t < 0 \end{cases}$, where n is a positive integer.
4. $g(t) = e^{-(t-a)^2/2b^2}$, for all t. That is, you should do the bilateral transform

Unified Engineering II **Spring 2005**

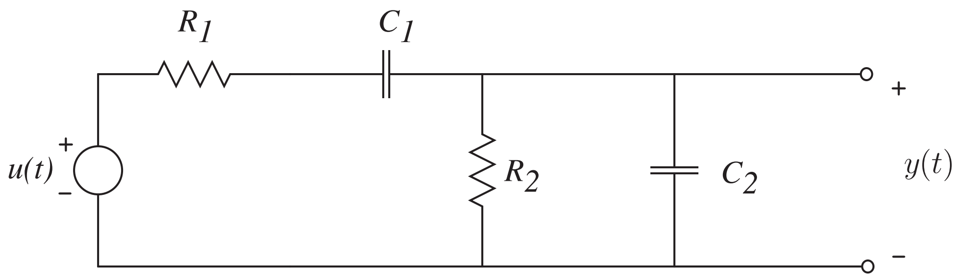
Problem S9 (Signals and Systems)

1. Find and plot the step response of the circuit below, using Laplace transform and impedance techniques.



Assume that $R = 2\ \Omega$, and $C = 0.1\,\mathrm{F}$. Note: This is a problem that's tricky to do using ordinary differential equation methods — Can you see why?

2. Find and plot the step response of the circuit below, using Laplace transform techniques.



Assume that $R_1 = R_2 = 2\,\Omega$, $C_1 = 0.2\,\mathrm{F}$, and $C_2 = 0.3\,\mathrm{F}$.

Unified Engineering II **Spring 2005**

Problem S10 (Signals and Systems)

This problem provides lots of practice using partial fraction expansions to determine inverse Laplace transforms. Please use the coverup method — it really is superior to other methods, and more reliable. Also, please check your answer, that is, verify that your expansion really is equivalent to the $G(s)$ given. For each of the following Laplace transforms, find the inverse Laplace transform.

1. $G(s) = \dfrac{3s^2 + 3s - 10}{s^2 - 4}, \quad \mathrm{Re}[s] > 2$

2. $G(s) = \dfrac{6s^2 + 26s + 26}{(s+1)(s+2)(s+3)}, \quad \mathrm{Re}[s] > -1$

3. $G(s) = \dfrac{4s^2 + 11s + 9}{(s+1)^2(s+2)}, \quad \mathrm{Re}[s] > -1$

4. $G(s) = \dfrac{4s^3 + 11s^2 + 5s + 2}{s^2(s+1)^2}, \quad \mathrm{Re}[s] > 0$

5. $G(s) = \dfrac{s^3 + 3s^2 + 9s + 12}{(s^2+4)(s^2+9)}, \quad \mathrm{Re}[s] > 0$