1. The differential equation is

$$\frac{d^2}{dt^2} y(t) + 5 \frac{d}{dt} y(t) + 6\, y(t) = u(t) = \sigma(t)$$

Find the homogeneous and particular solutions:

<u>homogeneous</u>:

Assume $y(t) = Y e^{st}$. Then

$$s^2 Y + 5s\,Y + 6Y = 0$$

$$\Rightarrow \quad s^2 + 5s + 6 = 0$$

$$\Rightarrow \quad (s+2)(s+3) = 0$$

$$\Rightarrow \quad s_1 = -2, \quad s_2 = -3$$

The homogeneous solution is therefore

$$y_h(t) = a e^{-2t} + b e^{-3t}$$

<u>particular</u>:

Since $u(t) = \sigma(t)$, $u(t) = 1 =$ constant for $t \geq 0$.
Therefore, assume

$$y_p(t) = c = \text{constant}$$

Plugging into the differential equation,

$$6c = 1 \quad \Rightarrow \quad c = 1/6$$

<u>total solution</u>:

The total solution is

$$y(t) = y_p(t) + y_h(t) = 1/6 + a e^{-2t} + b e^{-3t}$$

The ICs are $y(0)=0$, $y'(0)=0$. Therefore,

$$a+b = -1/6$$
$$-2a-3b = 0$$

Solving,

$$a = -1/2$$
$$b = 1/3$$

Therefore,

$$\boxed{\begin{aligned} g_s(t) &= \frac{1}{6} - \frac{1}{2}e^{-2t} + \frac{1}{3}e^{-3t}, \quad && t \geq 0 \\ &= 0 , && t < 0 \end{aligned}}$$

2. $$g(t) = \frac{d}{dt} g_s(t)$$
$$\begin{aligned} &= e^{-2t} - e^{-3t} \quad && t > 0 \\ &= 0 && t < 0 \\ &= 0 && t = 0 \end{aligned}$$

The last part is because $g_s(t)$ has no discontinuity at $t=0$. Therefore,

$$\boxed{g(t) = \sigma(t)\left[e^{-2t} - e^{-3t}\right]}$$

3. Since the input is an exponential, it makes sense to guess

$$y(t) = c\,e^{-2t}$$

If we plug this into the D.E, we obtain

$$4ce^{-2t} - 10ce^{-2t} + 6ce^{-2t} = e^{-2t}$$

$$\Rightarrow \quad 0 = e^{-2t}$$

But this is not possible. So our guess doesn't work.

As we'll see below, a better guess is

$$y_p(t) = c\,t\,e^{-2t}$$

4. $$y(t) = \int_0^t g(t-\tau)\,u(\tau)\,d\tau$$

$$= \int_0^t \left[e^{-2(t-\tau)} - e^{-3(t-\tau)}\right] e^{-2\tau}\,d\tau$$

$$= \int_0^t \left[e^{-2t} - 3e^{-3t+\tau}\right] d\tau$$

$$= e^{-2t}\int_0^t dt - 3e^{-3t}\int_0^t e^{\tau}\,d\tau$$

$$= e^{-2t}\cdot t - 3e^{-3t}\cdot\left(e^{t}-1\right)$$

Therefore,

$$\boxed{y(t) = \left[3e^{-2t} - 3e^{-3t} + te^{-2t}\right]\sigma(t)}$$

homogeneous

particular

So the response to an exponential is not always an exponential — sometimes it includes a <u>secular</u> term (one with a factor of t)

(a) $x(t) = e^{-\alpha t}\,\sigma(t)$

$h(t) = e^{-\beta t}\,\sigma(t)$

$$y(t) = h(t) * x(t)$$

$$= \int_{-\infty}^{\infty} h(t-\tau)\,x(\tau)\,d\tau$$

$$= 0, \qquad t < 0$$

$$= \int_0^t e^{-\beta(t-\tau)}\,e^{-\alpha\tau}\,d\tau, \qquad t \geq 0$$

$$= \int_0^t e^{-\beta t}\,e^{(\beta-\alpha)\tau}\,d\tau$$

If $\beta \neq \alpha$, then

$$y(t) = \int_0^t e^{-\beta t}\,e^{(\beta-\alpha)\tau}\,d\tau$$

$$= e^{-\beta t}\,\frac{1}{\beta-\alpha}\,e^{(\beta-\alpha)\tau}\Big|_{\tau=0}^{t}$$

$$= \frac{1}{\beta-\alpha}\,e^{-\beta t}\left(e^{(\beta-\alpha)t} - 1\right)$$

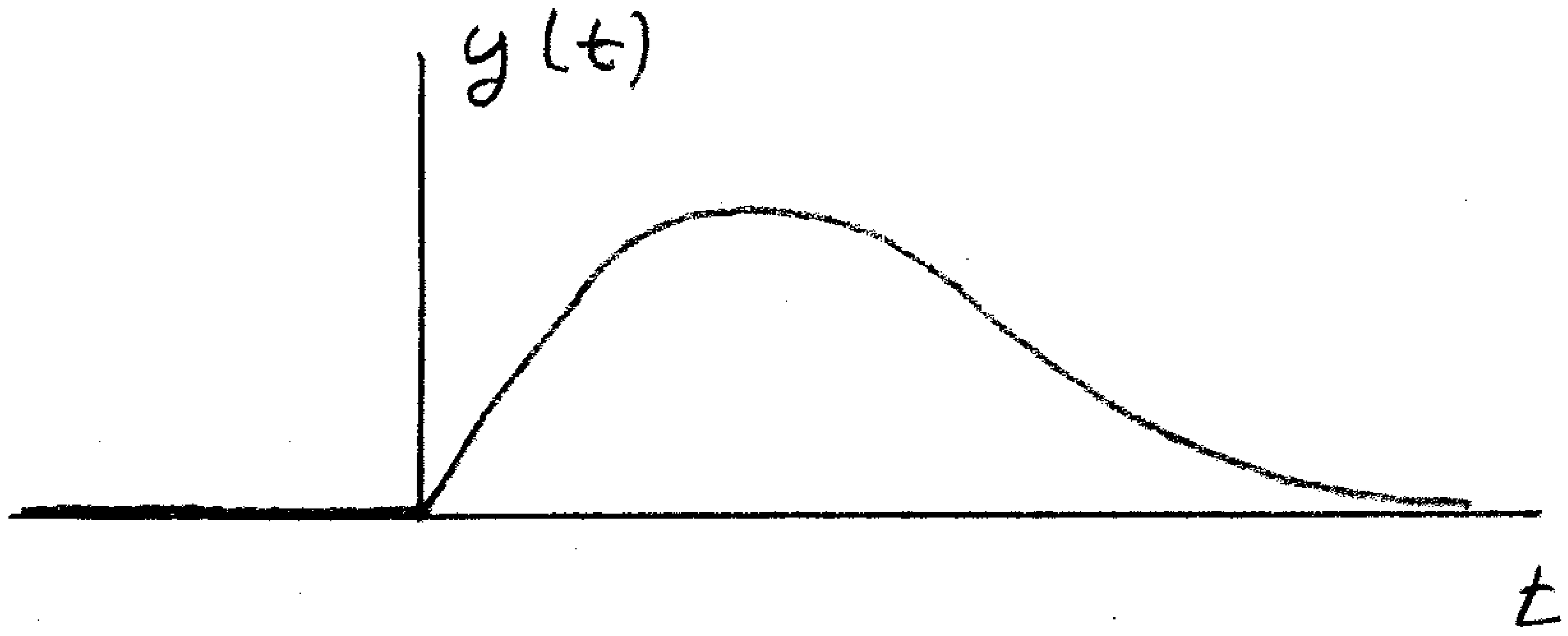
$$= \frac{1}{\beta-\alpha}\,e^{-\alpha t} - \frac{1}{\beta-\alpha}\,e^{-\beta t}, \qquad t \geq 0$$

If $\beta = \alpha$,

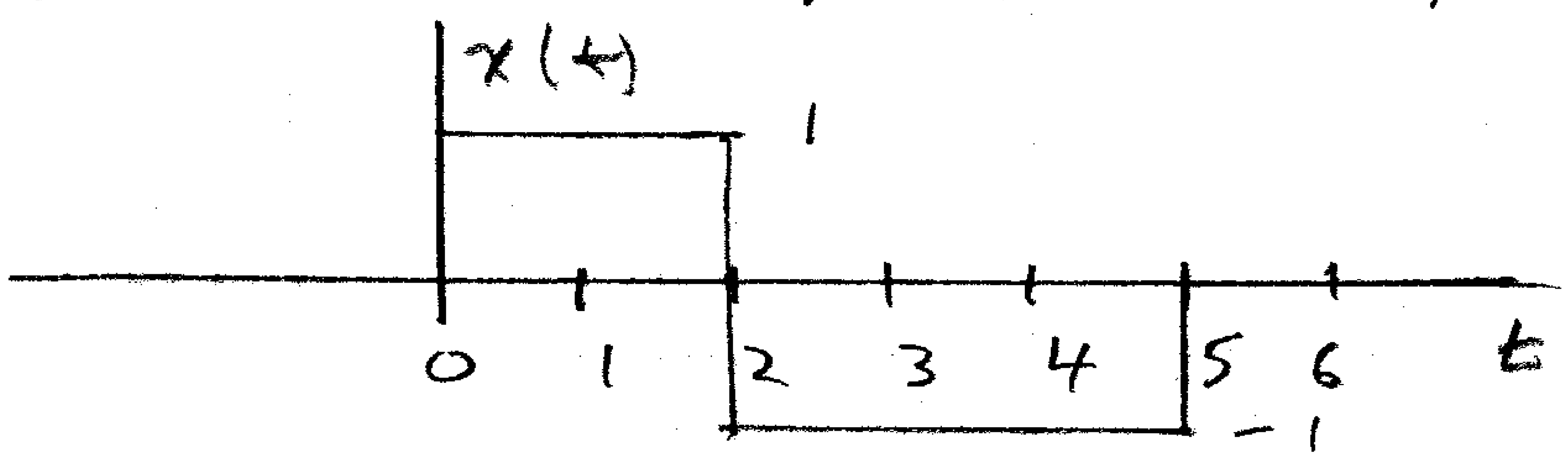
$$y(t) = \int_0^t e^{\alpha t}\,e^{0\tau}\,d\tau$$

$$= t\,e^{\alpha t}, \qquad t \geq 0$$

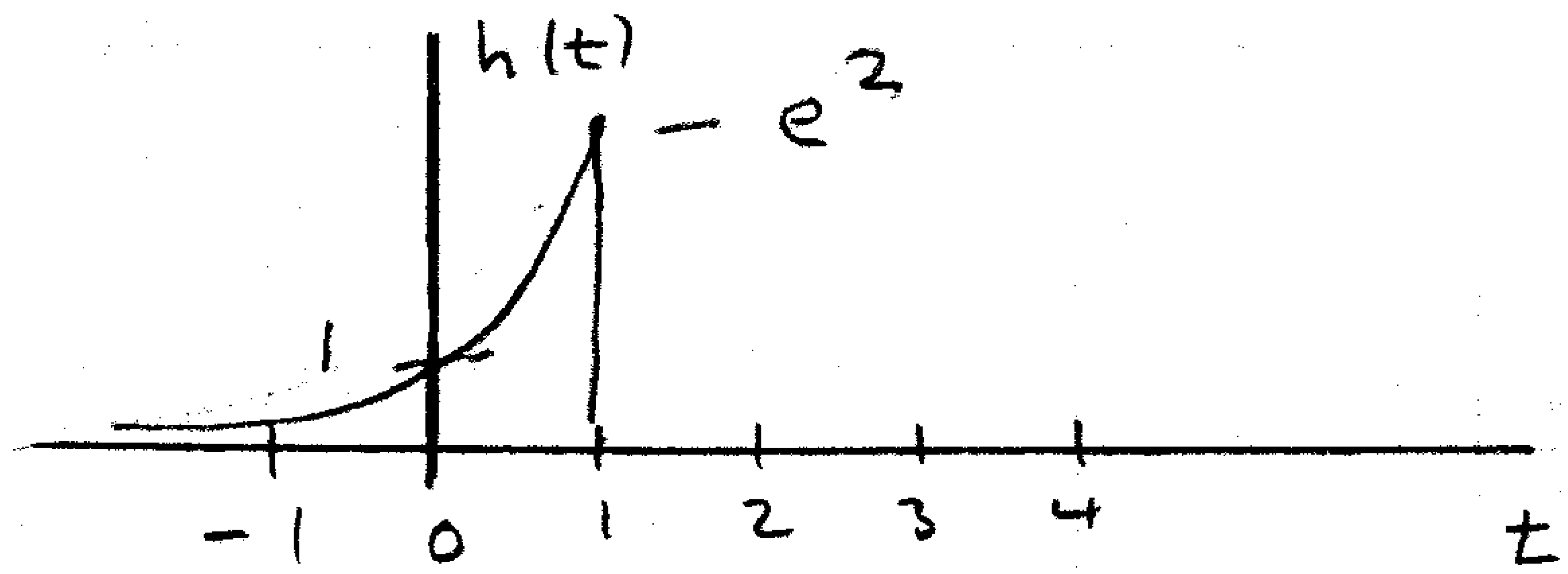
In either case, the result will look generally like



(6) $x(t) = r(t) - 2r(t-2) + r(t-5)$

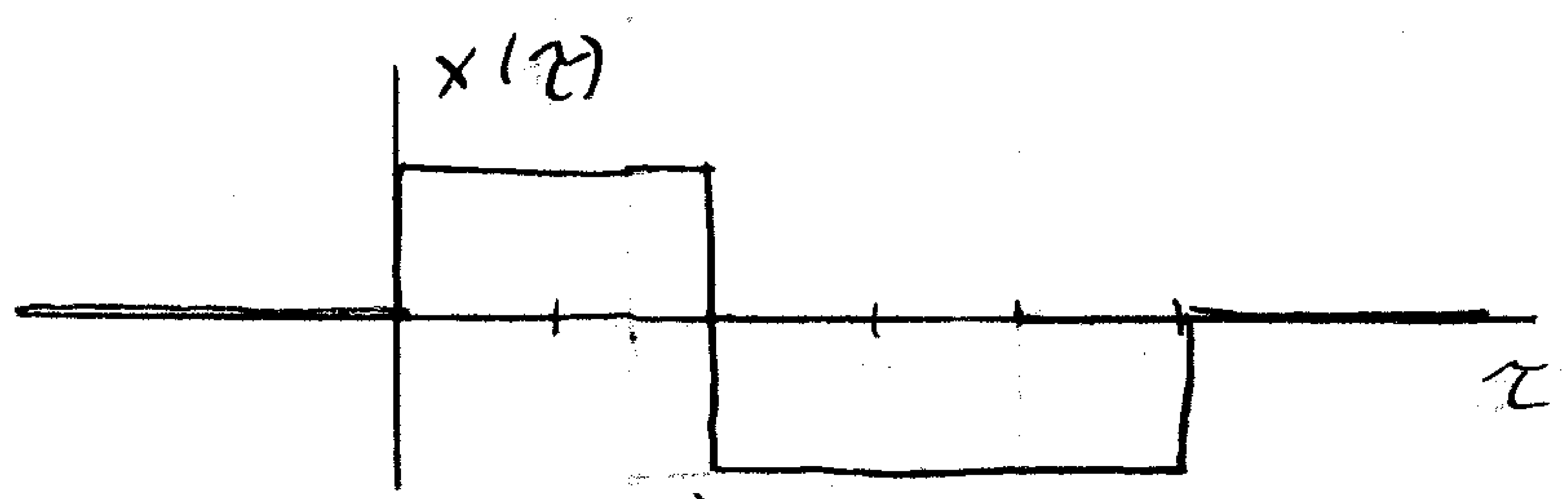


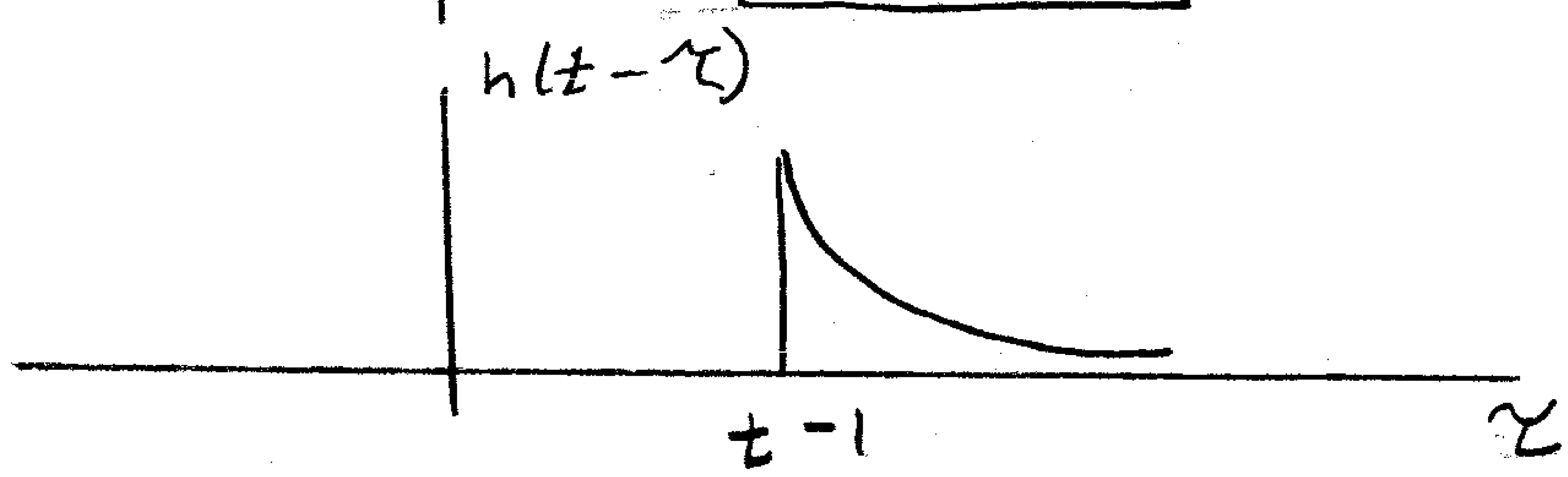
$h(t) = e^{2t}\, r(1-t)$



$y(t) = h(t) * x(t)$

Use flip & slide to get feel for answer:





Depending on the value of t, there are 4 cases:

<u>$t > 6$</u>: For this case, there is no overlap, so

$$y(t) = 0, \qquad t > 6$$

<u>$3 < t < 6$</u>: For this case,

$$y(t) = \int_{t-1}^{5} e^{2(t-\tau)} (-1)\, d\tau$$

$$= e^{2t} \int_{t-1}^{5} (-1) e^{-2\tau}\, d\tau$$

$$= e^{2t} \frac{1}{2} e^{-2\tau} \Big|_{\tau = t-1}^{5}$$

$$= e^{2t} \cdot \frac{1}{2} \left[e^{-10} - e^{-2(t-1)} \right]$$

<u>$1 < t < 3$</u>: For this case,

$$y(t) = \int_{t-1}^{2} e^{2(t-\tau)} (1)\, d\tau + \int_{2}^{5} e^{2(t-\tau)} (-1)\, d\tau$$

$$= -\frac{1}{2} e^{2t} e^{-2\tau} \Big|_{\tau = t-1}^{2} + \frac{1}{2} e^{2t} e^{-2\tau} \Big|_{\tau = 2}^{5}$$

$$= \frac{1}{2} e^{2t} \left(e^{-2(t-1)} - e^{-4} \right) + \frac{1}{2} e^{2t} \left(e^{-10} - e^{-4} \right)$$

<u>$t < 1$</u>: For this case,

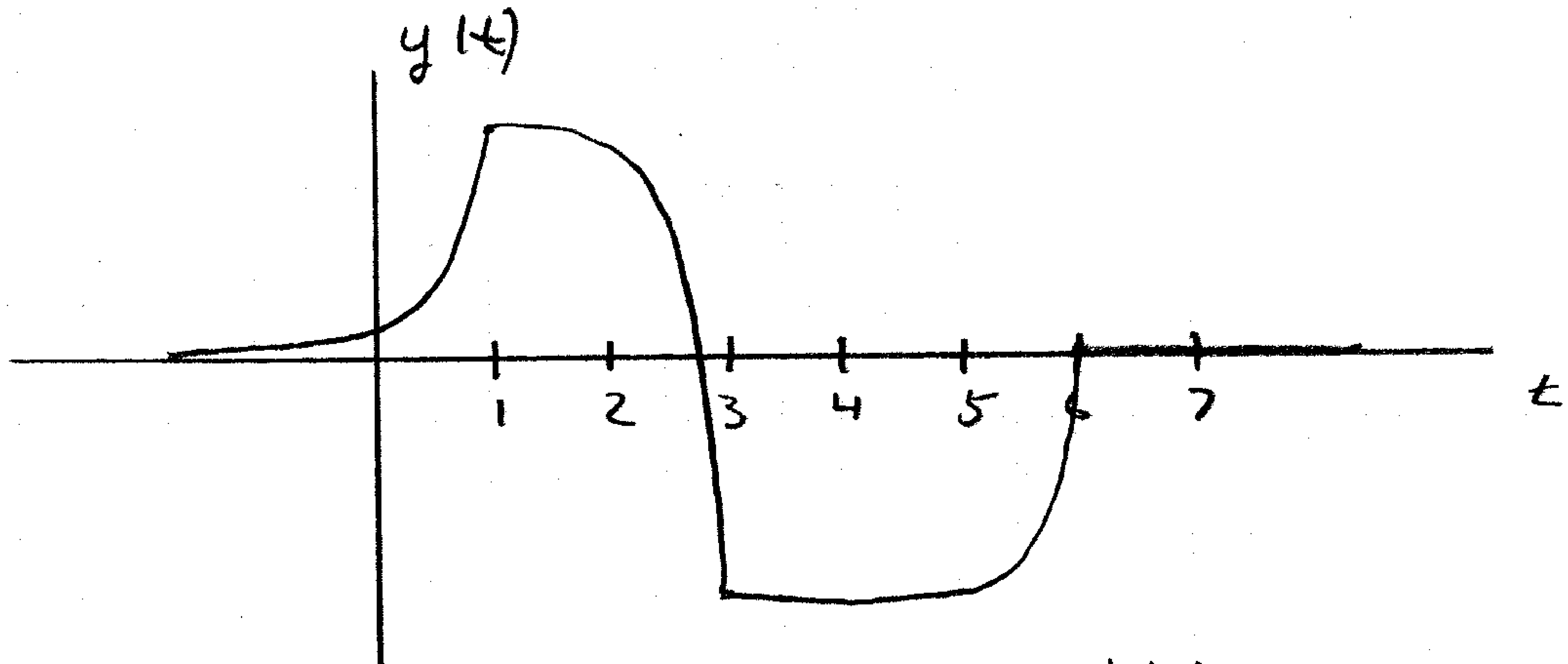
$$y(t) = \int_{0}^{2} e^{2(t-\tau)} (1)\, d\tau + \int_{2}^{5} e^{2(t-\tau)} (-1)\, d\tau$$

$$= \frac{1}{2} e^{2t}\left(e^{0} - e^{-4}\right) + \frac{1}{2} e^{2t}\left(e^{-10} - e^{-4}\right)$$

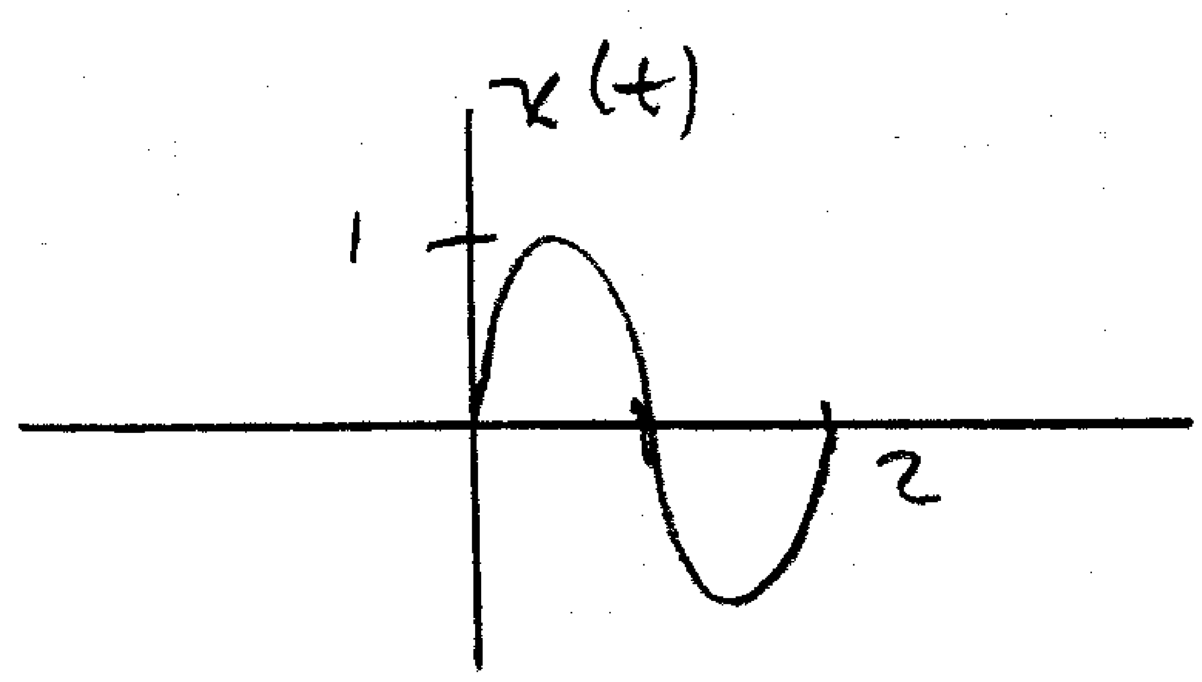
Simplifying,

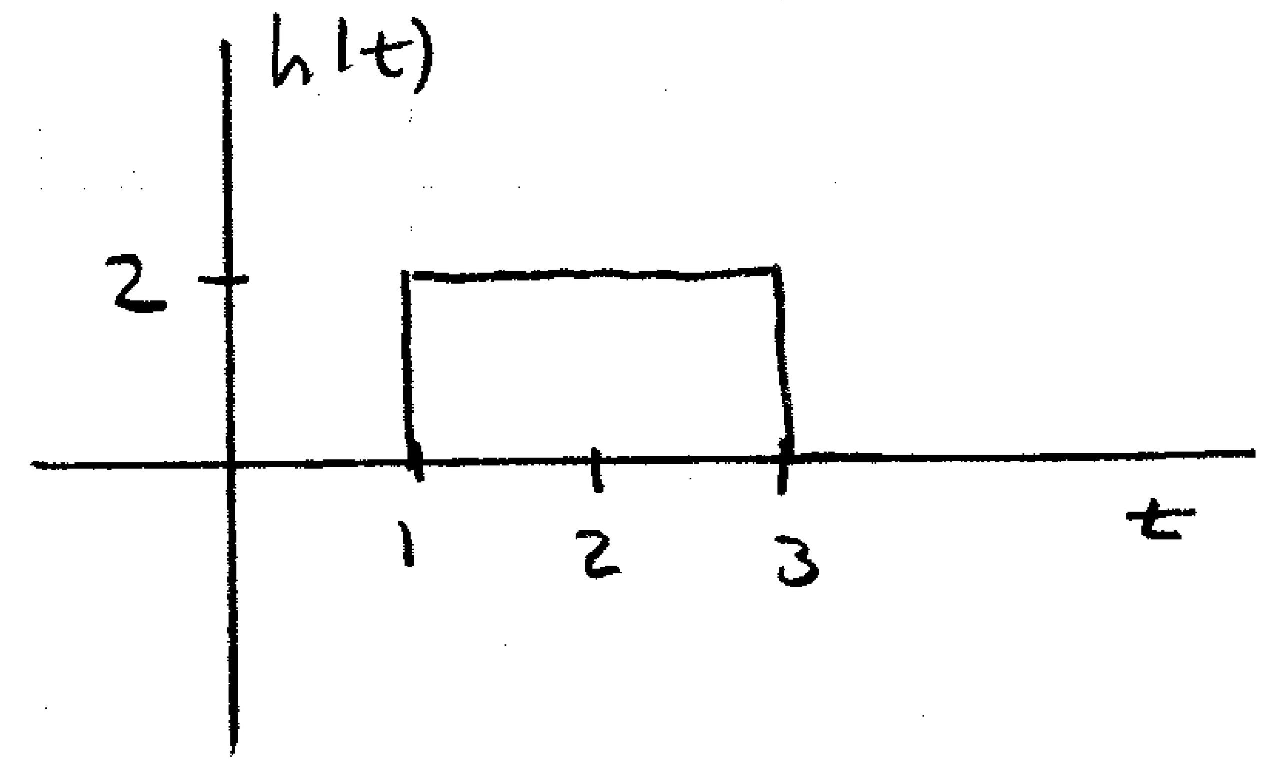
$$y(t) = \begin{cases} \frac{1}{2} e^{2t}\left(1 - 2e^{-4} + e^{-10}\right), & t < 1 \\ \frac{1}{2} e^{2} + \frac{1}{2} e^{2t}\left(e^{-10} - 2e^{-4}\right), & 1 < t < 3 \\ \frac{1}{2} e^{2t-10} - \frac{1}{2} e^{2}, & 3 < t < 6 \\ 0, & t > 6 \end{cases}$$

Sketch of $y(t)$:

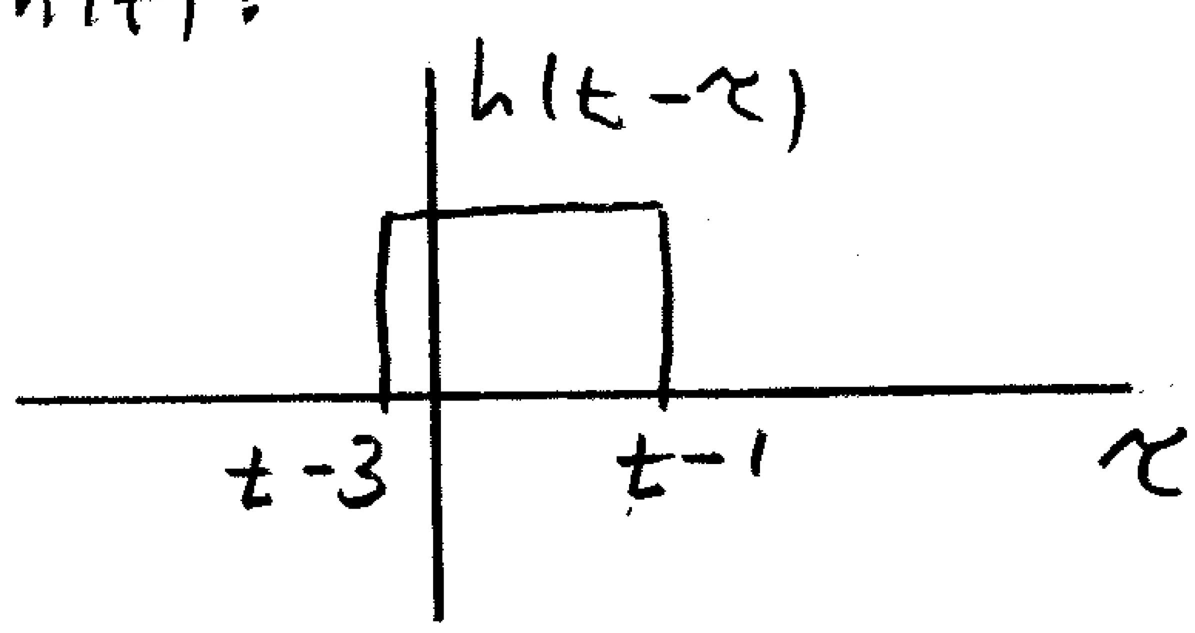


(c)





Flip $h(t)$:



There are 4 cases:

$t < 1$: In this case, there is no overlap, so

$$y(t) = 0$$

$1 < t < 3$: $y(t) = \int_0^{t-1} 2 \cdot \sin \pi\tau \, d\tau$

$$= \frac{-2}{\pi} \cos \pi\tau \Big|_{\tau=0}^{t-1}$$

$$= \frac{-2}{\pi}\left[\cos \pi(t-1) - 1\right]$$

$$= \frac{2}{\pi}\left[1 + \cos \pi t\right]$$

$3 < t < 5$: $y(t) = \int_{t-3}^{2} 2 \sin \pi\tau \, d\tau$

$$= -\frac{2}{\pi} \cos \pi\tau \Big|_{\tau=t-3}^{2}$$

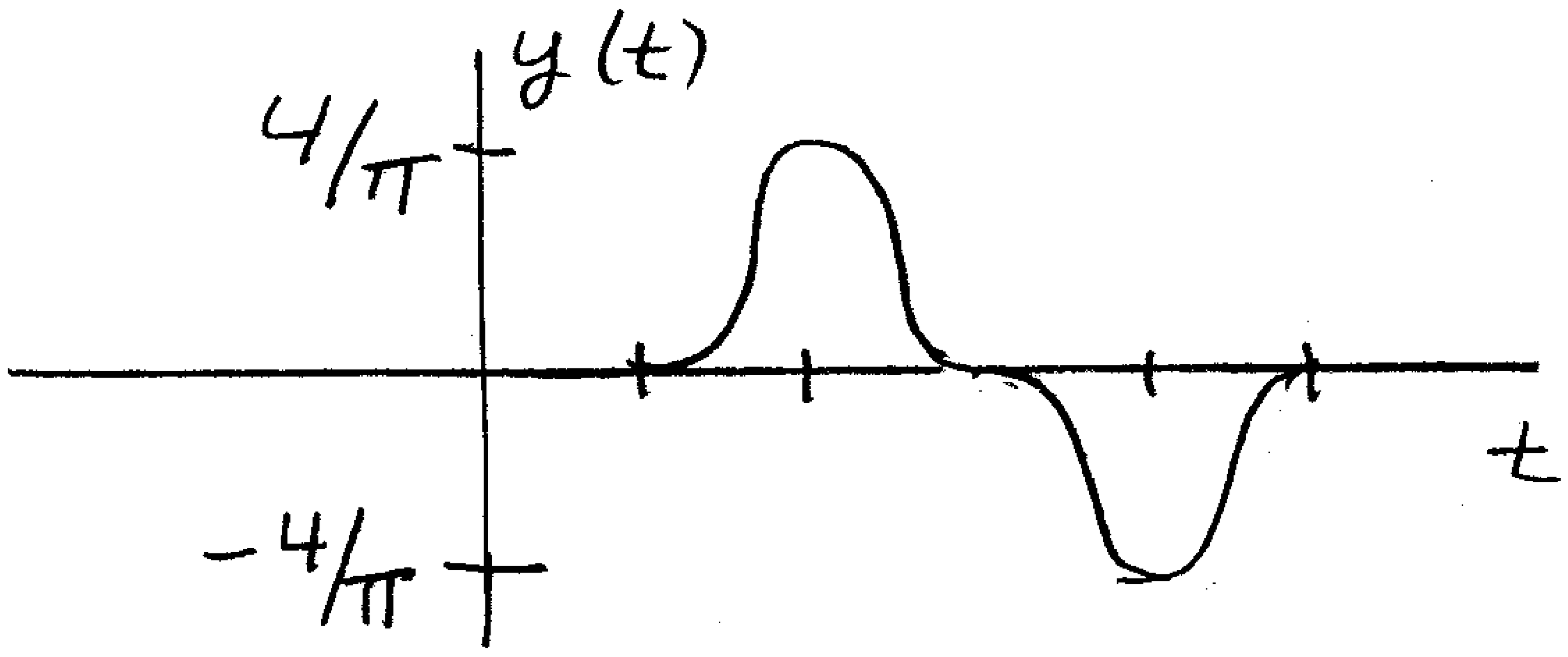
$$= -\frac{2}{\pi}\left[\cos \pi 2 - \cos \pi(t-3)\right]$$

$$= -\frac{2}{\pi}\left[1 + \cos \pi t\right]$$

$t > 5$: There is no overlap, so $y(t) = 0$.

Therefore,

$$y(t) = \begin{cases} \frac{2}{\pi}(1 + \cos \pi t) & 1 < t < 3 \\ -\frac{2}{\pi}(1 - \cos \pi t) & 3 < t < 5 \\ 0 & \text{else} \end{cases}$$



(d) $y(t) = h(t) * x(t)$ $\quad\quad x(t) = a + bt$

$= x(t) * h(t)$ $\quad\quad h(t) = \frac{4}{3}\left[r(t) - r(t-1)\right] - \frac{1}{3}\delta(t-2)$

$$y(t) = \int_0^1 \frac{4}{3}\left[a + b(t-\tau)\right] d\tau$$

$$+ \int -\frac{1}{3}\delta(\tau-2)\left[a + b(t-\tau)\right] dt$$

$$= \frac{4}{3}\left[a + bt\right]\tau \Big|_{\tau=0}^{1} - \frac{2}{3} b\tau^2 \Big|_{\tau=0}^{1}$$

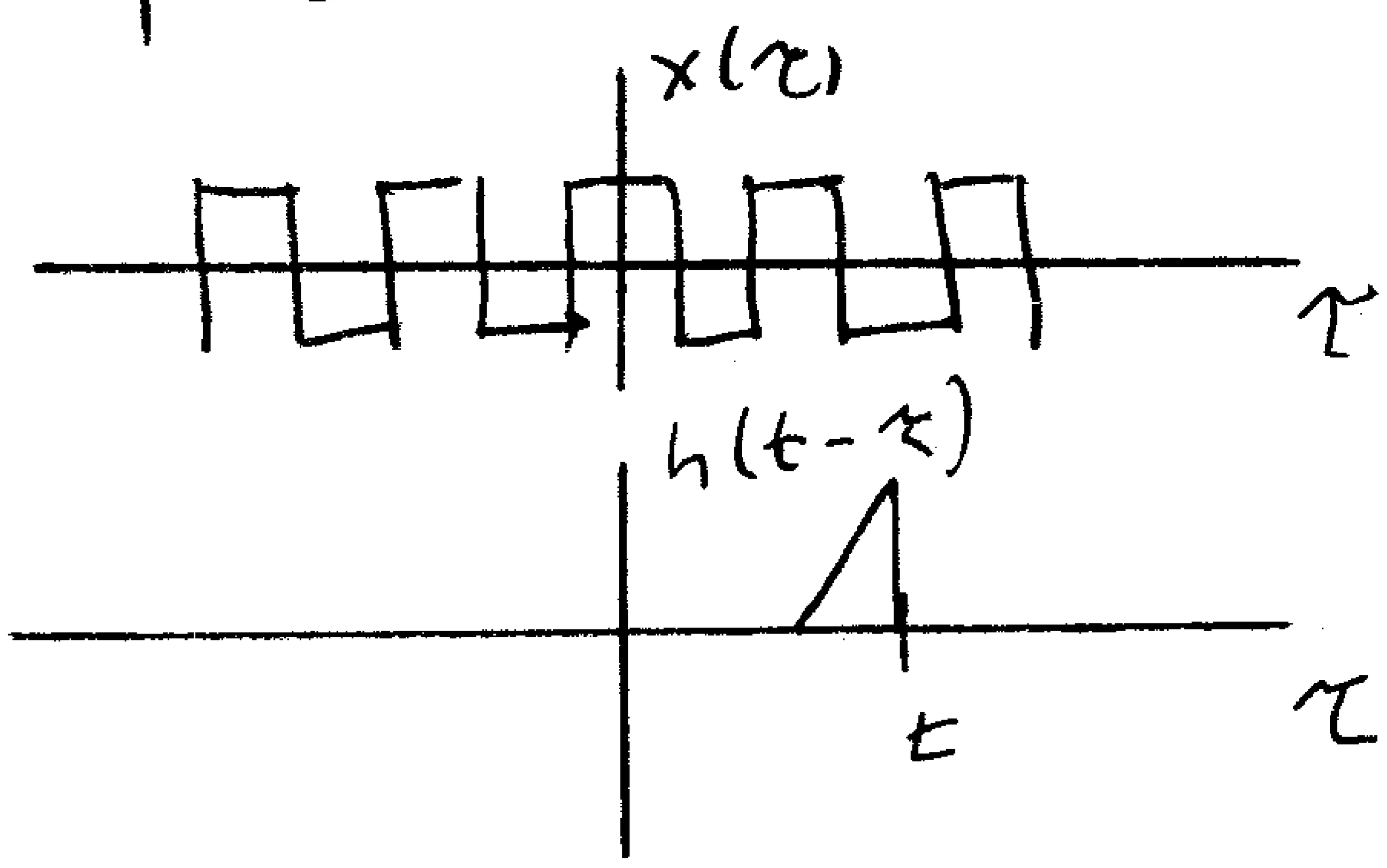
$$- \frac{1}{3}\left[a + b(t-2)\right]$$

$$= \frac{4}{3}\left[a + bt\right] - \frac{2}{3}b - \frac{1}{3}a + \frac{2}{3}b - bt\Big]$$

$$= a + bt$$

So $y(t) = x(t)$!!!

(e) Flip & Slide:



when $h(t)$ overlaps a positive pulse,

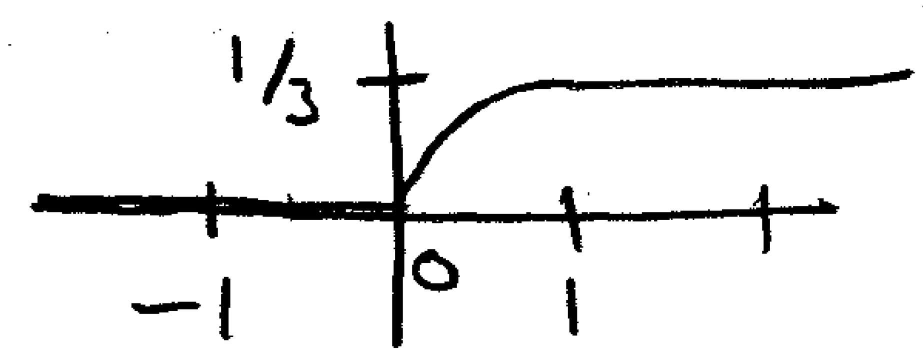
$$y(t) = \int 1 \cdot h(t-\tau)\, d\tau = 1/3$$

when $h(t)$ overlaps a negative pulse

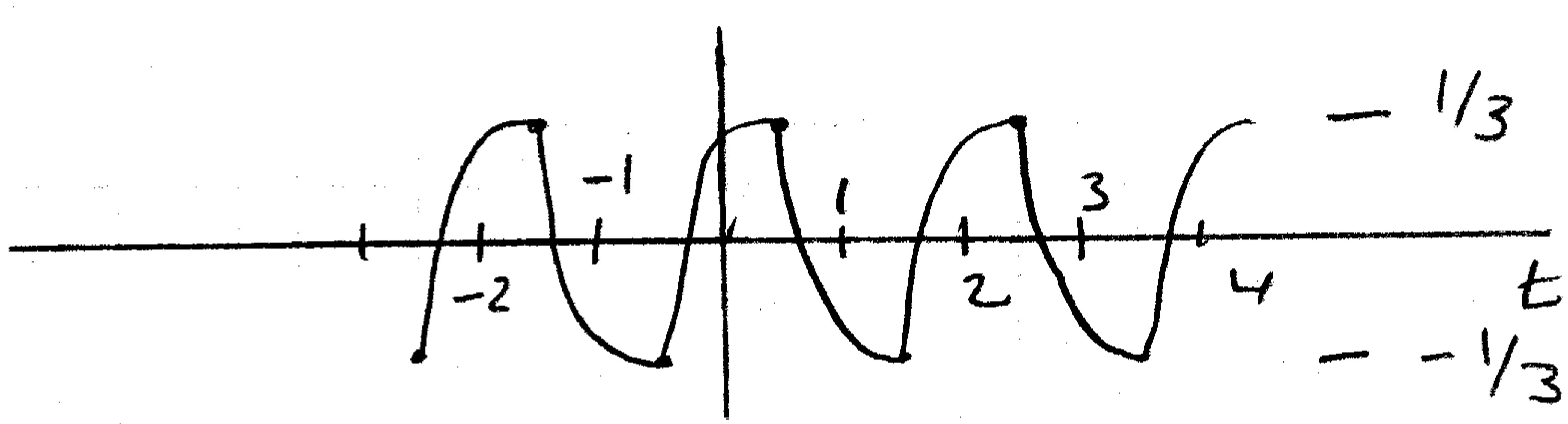
$$y(t) = -1/3$$

If $h(t)$ is convolved with a step, $\sigma(t)$, the result is

$$h(t) * \sigma(t) =$$

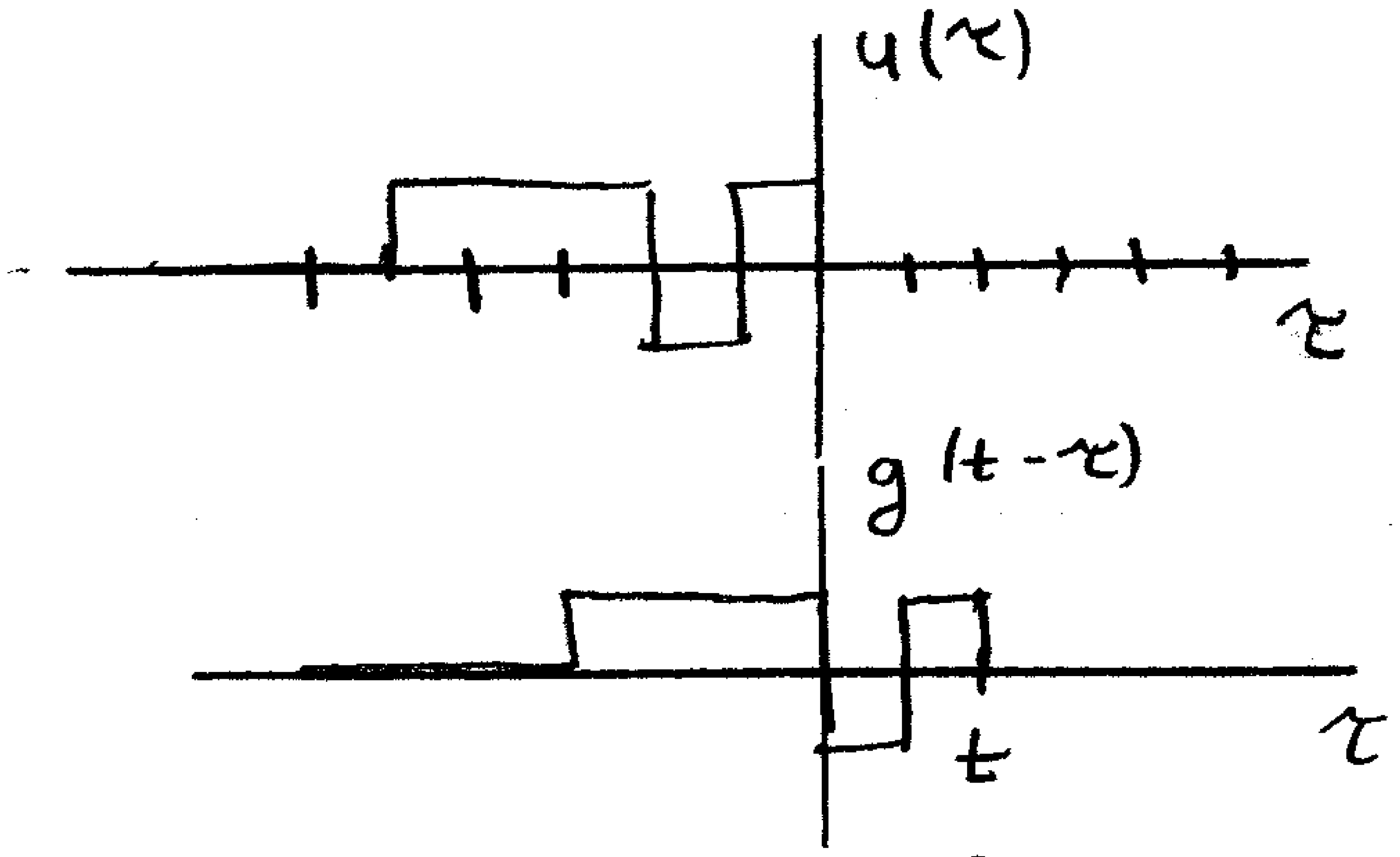


Therefore, $h(t) * x(t)$ should look like



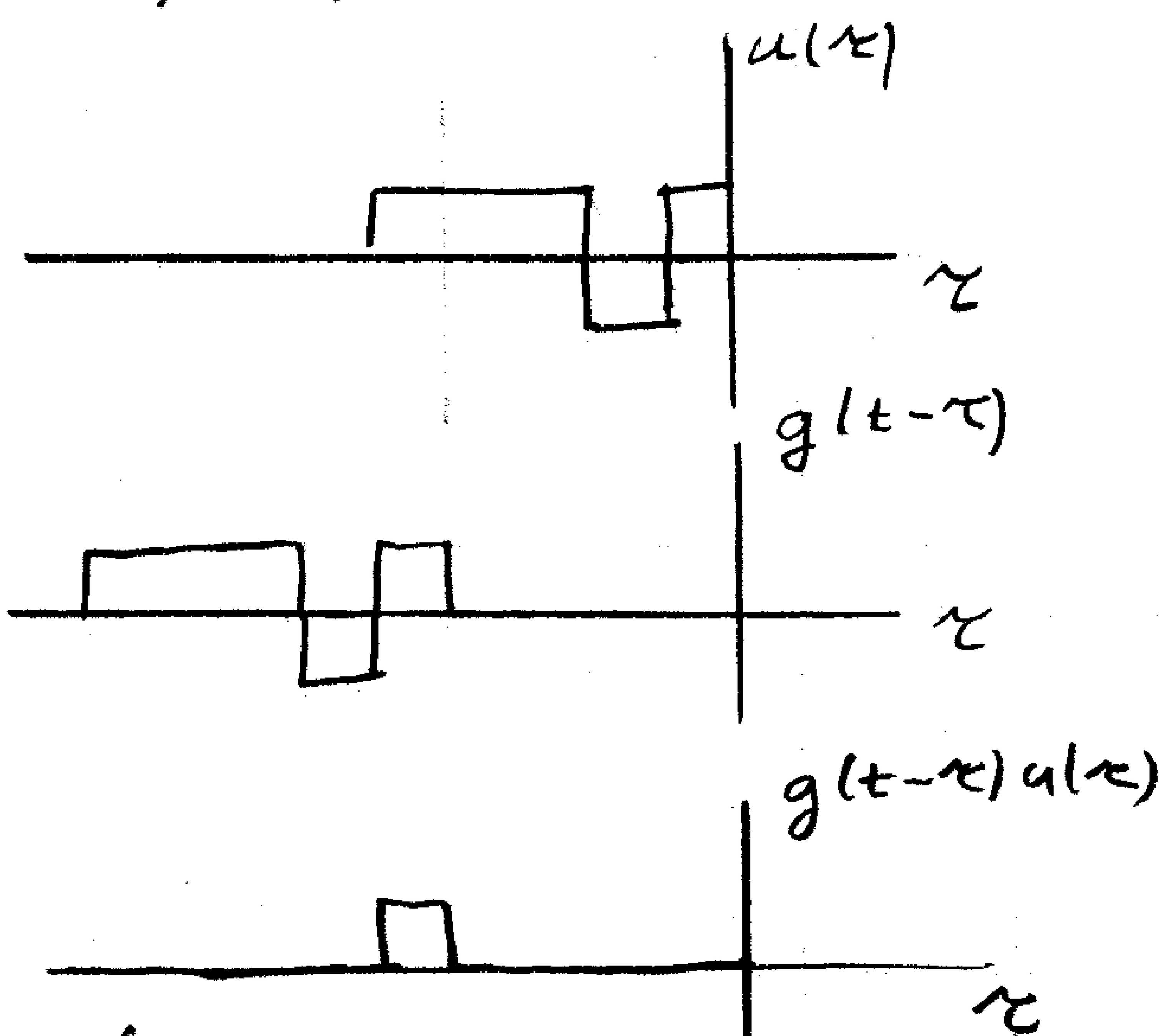
1. Because $g(t)$ & $u(t)$ are piecewise constant, $y(t)$ will be continuous and piecewise linear. We can find $y(t)$ at the "corners" by evaluating at $t =$ integer, since the corners of $g(t)$ & $u(t)$ occur at the integers.

So do flip & slide:



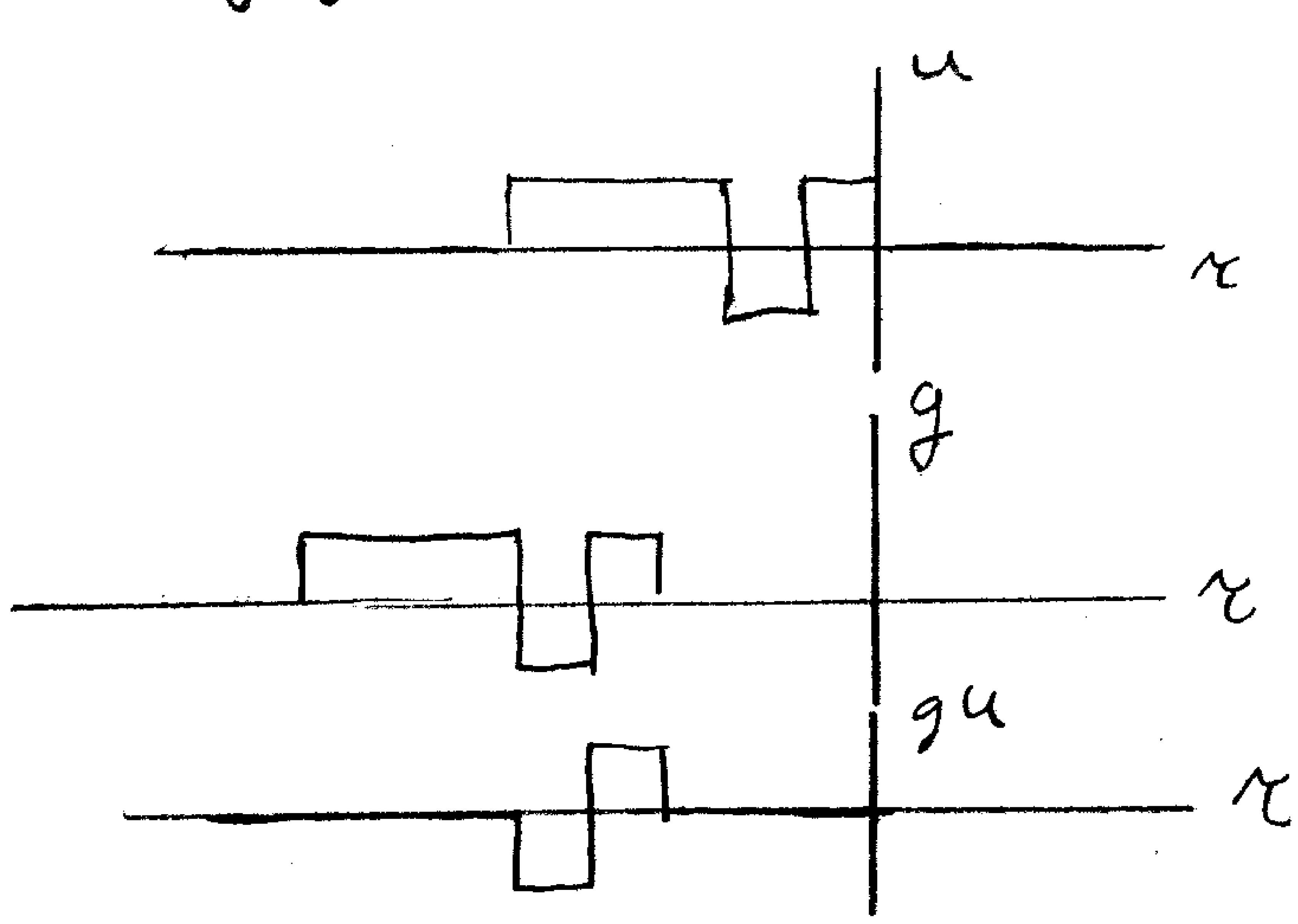
There is no overlap for $t < -5$ or $t > 5$. So do $t = -4, -3, -2, \ldots 4$

$t = -4$:



so $y(-4) = \int g(t-\tau)\,u(\tau)\,d\tau = 1$

$t = -3$:

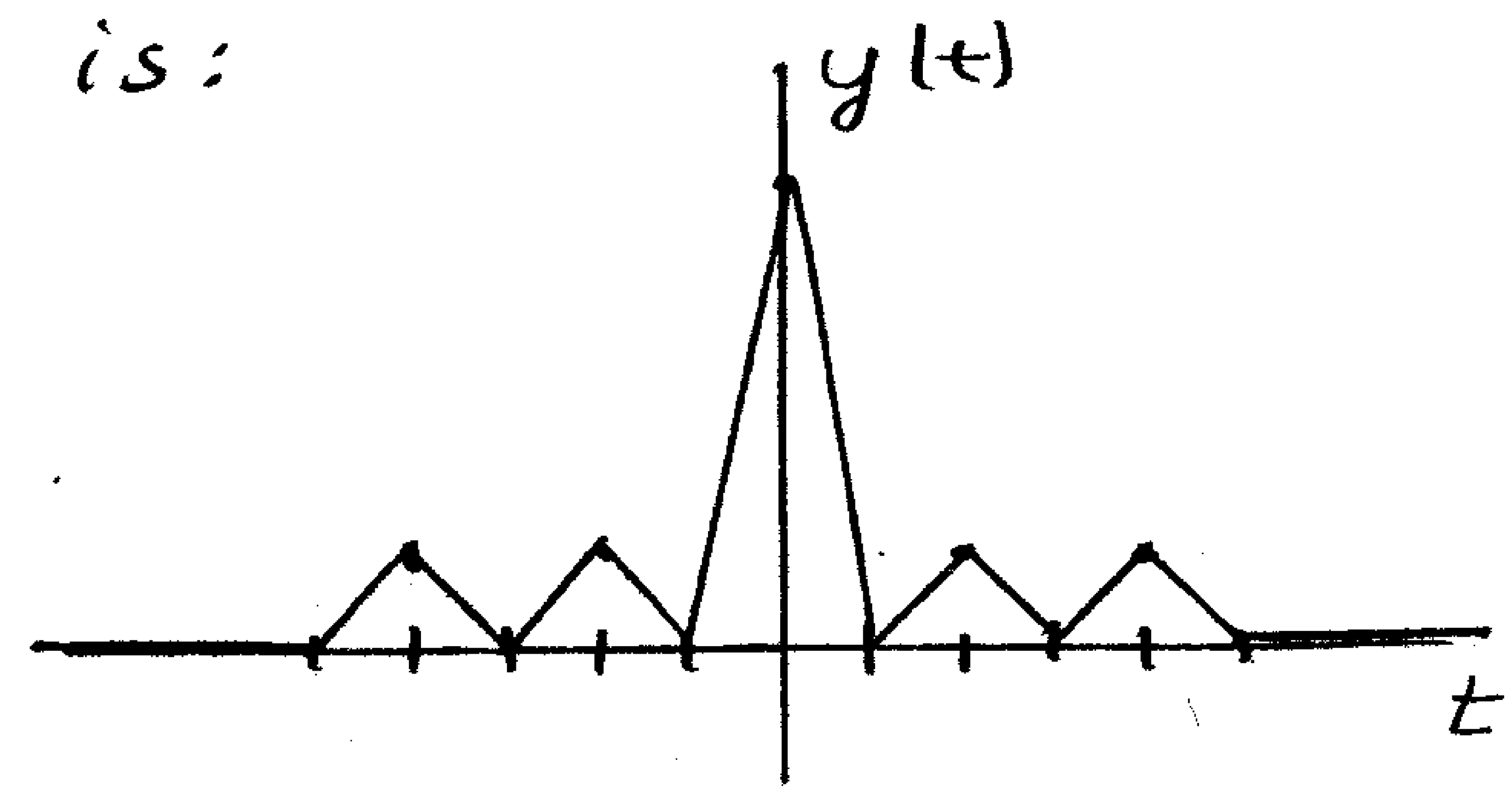


So $y(-3) = \int g(t-\tau)u(\tau)\,d\tau = 0$

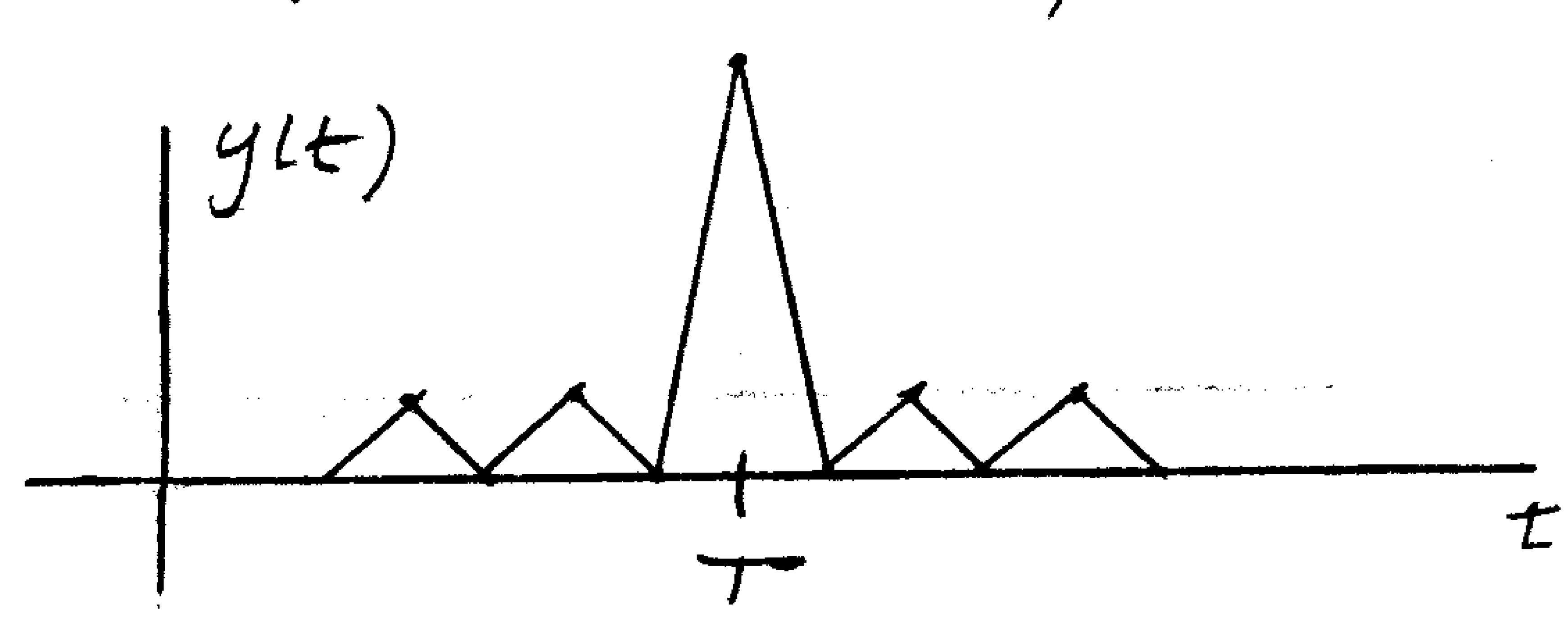
Continuing in this fashion, we have

t	$y(t)$
-4	1
-3	0
-2	1
-1	0
0	5
1	0
2	1
3	0
4	1
5	0

So $y(t)$ is:



2. By linearity and time invariance, delaying $u(t)$ by T will simply delay $y(t)$, so the convolution $g(t) * u(t-T)$ is as above, shifted right by T.



3. T is easily identified as the time at which the max of $y(t)$ occurs.

1. $$G(s) = \int_0^\infty t e^{-at} e^{-st} dt$$

$$= \int_0^\infty t e^{-(s+a)t} dt$$

Integrate by parts:

$$U = t \qquad \Rightarrow \qquad dU = dt$$

$$dv = e^{-(s+a)t} dt \qquad V = \frac{-1}{s+a} e^{-(s+a)t}$$

Therefore,

$$G(s) = UV - \int V dU$$

$$= \frac{-t}{s+a} e^{-(s+a)t} \Big|_0^\infty + \frac{1}{s+a} \int_0^\infty e^{-(s+a)t} dt$$

$$= 0 + \frac{1}{(s+a)^2} \quad \text{if} \quad Re[s] > -a$$

$$\boxed{G(s) = \frac{1}{(s+a)^2}, \quad Re[s] > -a}$$

2. $$G(s) = \int_0^\infty \underbrace{t^2}_{U} \underbrace{e^{-at} e^{-st} dt}_{dv}$$

$$dU = 2t\, dt, \quad v = \frac{-1}{s+a} e^{-(s+a)t}$$

Integrating by parts,

$$G(s) = \underbrace{\frac{-t^2}{s+a} e^{-(s+a)t} \Big|_0^\infty}_{=0,\ Re[s] > -a} + \frac{1}{s+a} \int_0^\infty 2t\, e^{-(s+a)t} dt$$

The second term above is known from part (1) above. Therefore,

$$\boxed{G(s) = \frac{2}{(s+a)^3}, \quad Re[s] > -a}$$

3. The pattern should be clear. In general,

$$\boxed{\mathcal{L}[t^n e^{-at}] = \frac{n!}{(s+a)^{n+1}}, \quad Re[s] > -a}$$

4. For

$$f(t) = e^{-(t-a)^2/2b^2}, \quad \text{for all } t$$

the LT is

$$F(s) = \int_{-\infty}^{\infty} e^{-(t-a)^2/2b^2} e^{-st}\, dt$$
$$= \int_{-\infty}^{\infty} e^{-(t-a)^2/2b^2 - st}\, dt$$

The exponent is

$$-\frac{(t-a)^2}{2b^2} - st = -\frac{t^2}{2b^2} + \frac{at}{b^2} - \frac{a^2}{2b^2} - st$$
$$= -\frac{t^2}{2b^2} + \left(\frac{a}{b^2} - s\right)t - \frac{a^2}{2b^2}$$

Complete the square to obtain

$$\text{exponent} = -\frac{1}{2b^2}\left(t - [a - sb^2]\right)^2 + \frac{s^2b^2}{2} - as$$

Therefore,

$$F(s) = \int_{-\infty}^{\infty} e^{-(t-(a-sb^2))^2/2b^2} e^{s^2b^2/2 - as}\, dt$$

$$G(s) = e^{s^2b^2/2 - as} \int_{-\infty}^{\infty} e^{-[t-(a-sb^2)]^2/2b^2}$$

The integral above has integrand which is a Gaussian. Therefore,

$$G(s) = e^{s^2b^2/2 - as} \cdot \underbrace{\sqrt{2\pi} \cdot b}_{\text{Factors required to normalize Gaussian}}$$

The integral converges for all s, because the exponent is dominated by $-t^2/2b^2$. Therefore,

$$\boxed{F(s) = \sqrt{2\pi}\, b\, e^{s^2b^2/2 - as}, \quad \text{all } s}$$

No. 5505
Engineer's Computation Pad

CP5_6
The problems in this problem set cover lectures C5 and C6.

1.
a. What are doubly linked lists? What is the record declaration for a node in a doubly linked list?

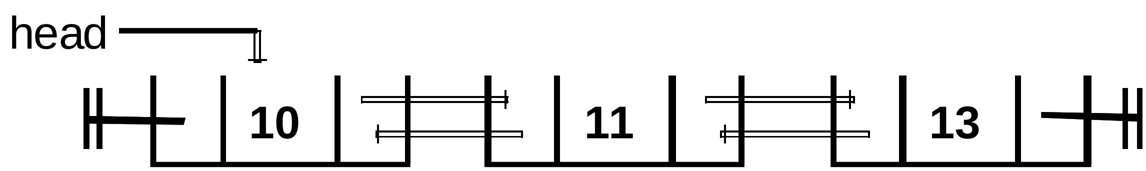


Figure 1. Sample Doubly Linked List

Doubly linked lists have two pointers instead of the single pointer seen in singly linked lists. The pointers point to both the previous node in the list as well as the next node in the list.

```
type Listnode is
  record
    Element : Elementtype;
    Next    : Listptr;
    Prev    : Listptr;  -- this is the change made to singly linked lists
  end record;
```

b. Write an algorithm to insert a node into a sorted doubly linked list. Use a diagram to show the sequence of operations that have to be performed to carry out the insertion step.

Hint: Extend the approach used in class/ notes for singly linked lists.

Preconditions:

1. User passes the list (called List) and the element to be inserted (called Element) to the insert procedure
2. List is already sorted

Postconidtions:

1. Procedure returns the list with the element inserted in the correct position
2. List remains sorted

Algorithm:

Create three temporary Listptrs *Current*, *Previous* and *NewNode*

Previous := null

Current := List.Head;

NewNode := new Listnode;

NewNode.Element := Element

```
Loop
    exit when Current = Null
    exit when  Current.Element > Element
    Previous := Current;
    Current := Current.Next;

NewNode. Next := Current;
NewNode.Prev:= Previous;
If Previous = null
    L.Head := NewNode
else
    Previous.Next := NewNode
    If Current /= null
        Current.Prev := NewNode;
Return List
```

Consider the doubly linked list shown in Figure 1. The insertion of a node with element 12 in the list, the insert operation is shown in Figures 2 and 3.

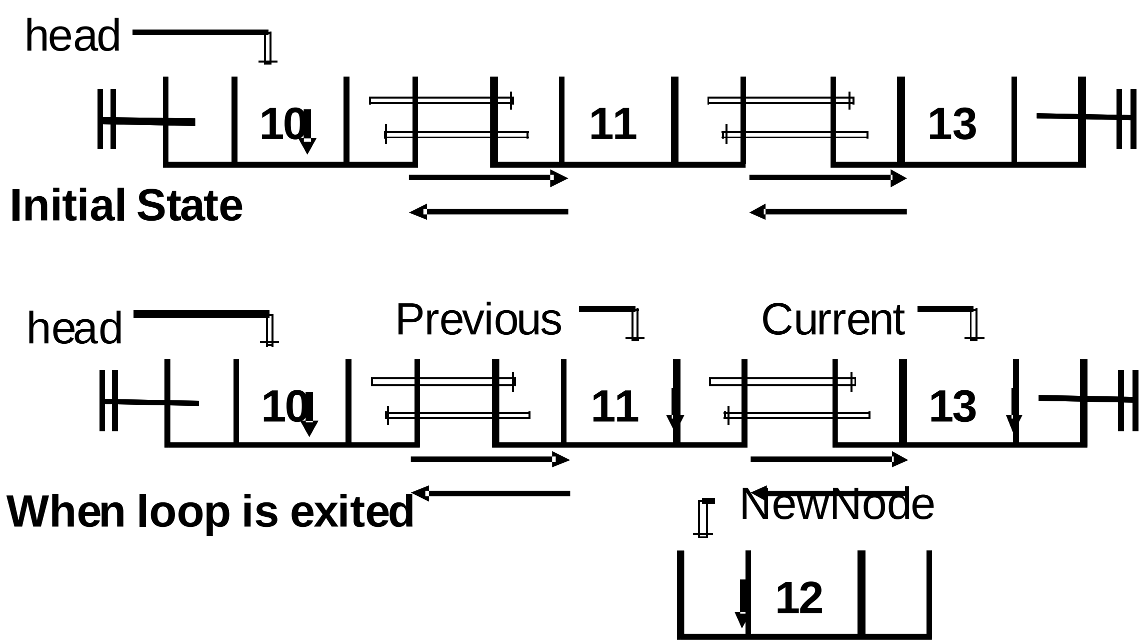
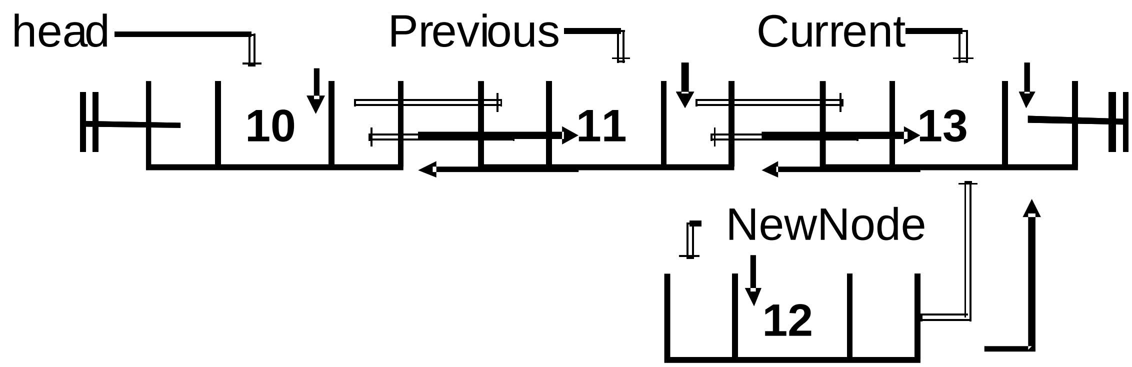
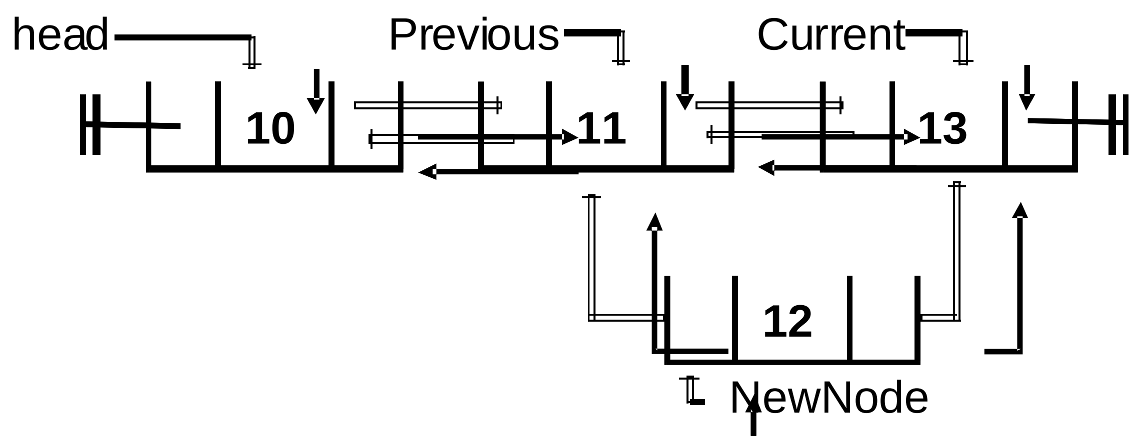


Figure 2. Prior to Insertion

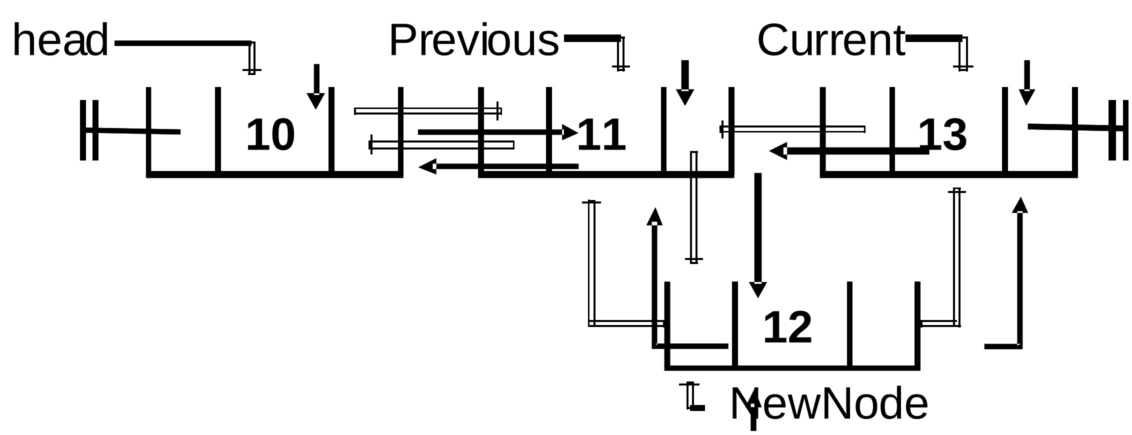
NewNode.Next := Current



NewNode.Prev := Previous



Previous.Next := NewNode



Current.Prev := NewNode

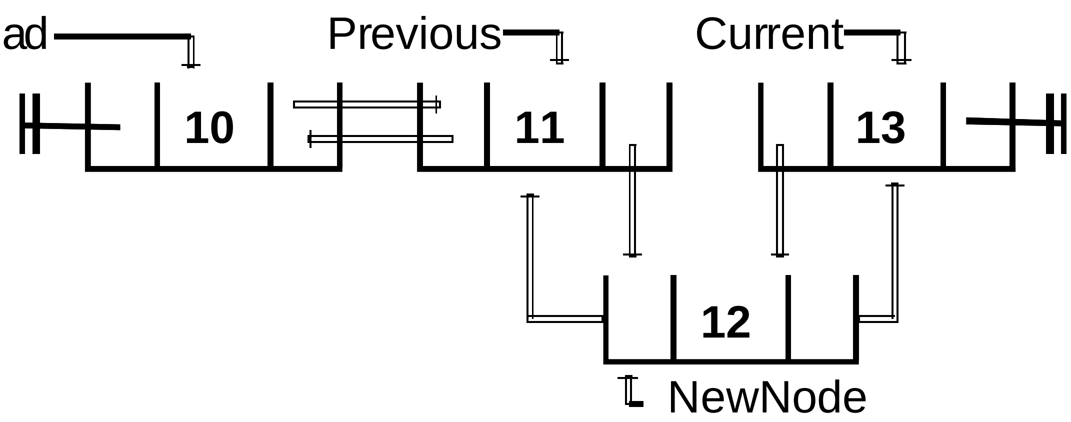


Figure 3. After Insertion Operation

c. Implement your algorithm as an Ada95 program.

Package Specification

GNAT 3.15p (20020523) Copyright 1992-2002 Free Software Foundation, Inc.

Checking: c:/docume~1/jayaka~1/mydocu~1/16070/code/doubly_linked_list.ads (source file time stamp: 2004-03-31 20:46:40)

```
---------------------------------------------------------------------
-- Specification for  doubly-linked lists
-- Specified: Jayakanth Srinivasan
-- Last Modified: February 11, 2004
---------------------------------------------------------------------

package Doubly_Linked_List is

   subtype Elementtype is Integer;

   type Listnode;
   type Listptr is access Listnode;
   type Listnode is
      record
         Element : Elementtype;
         Next    : Listptr;
         Prev    : Listptr;      -- this is the change made to singly linked lists
      end record;

   type List is
      record
         Head : Listptr;
      end record;




```

```
   procedure Makeempty (
         L : in out List );
   -- Pre:    L is defined
   -- Post:   L is empty

   function Isempty (
         L : in     List )
     return Boolean;
   -- Pre:    L is defined
   -- Post:   returns True if L is empty, False otherwise

   procedure Display (
         L : in     List );
   -- Pre:  L may be empty
   -- Post: displays the contents of L's Element fields, in the
   --   order in which they appear in L

   procedure Initialize (
         L : in out List );

   -- Pre: L may be empty
   -- Post: Elements inserted into the list at correct position
   procedure Insert_In_Order (
         L       : in out List;
         Element : in     Elementtype );

end Doubly_Linked_List;
```

54 lines: No errors

Package Implementation

GNAT 3.15p (20020523) Copyright 1992-2002 Free Software Foundation, Inc.

Compiling: c:/docume~1/jayaka~1/mydocu~1/16070/code/doubly_linked_list.adb (source file time stamp: 2004-03-31 20:48:14)

```
---------------------------------------------------------------------
-- Implementation for  doubly-linked lists
-- Programmer: Jayakanth Srinivasan
-- Last Modified: Feb 11, 2004
---------------------------------------------------------------------

with Ada.Text_Io;
with Ada.Integer_Text_Io;
with Ada.Unchecked_Deallocation;

use Ada.Text_Io;
use Ada.Integer_Text_Io;

package body Doubly_Linked_List is
   -- create an instance of the free procedure
   procedure Free is
   new Ada.Unchecked_Deallocation(Listnode, Listptr);
   -- check if list is empty. List.Head will be null
   function Isempty (
         L : in     List )
     return Boolean is
```

```
begin
  if L.Head = null then
    return True;
  else
    return False;
  end if;

end Isempty;

-- free all allocated memory at the end of the program
procedure Makeempty (
    L : in out List ) is
  Temp : Listptr;

begin
  loop
    exit when Isempty(L);
    Temp := L.Head;
    L.Head := Temp.Next;
    Free(Temp);
  end loop;
  L.Head := null;
end Makeempty;

-- initialize the list by setting the head pointed to null
procedure Initialize (
    L : in out List ) is
begin
  L.Head := null;
end Initialize;

-- displays the contents of the list
procedure Display (
    L : in     List ) is
  Temp : Listptr;
begin
  -- set the pointer to the head of the node
  Temp:= L.Head;
  while Temp /= null loop
    Put(Temp.Element);
    Put(" , ");
    -- move pointer to the next node
    Temp :=Temp.Next;
  end loop;
  New_Line;
end Display;

-- insert elements in ascending order
--  this procedure added :)
procedure Insert_In_Order (
    L       : in out List;
    Element : in     Elementtype ) is
  Current,
  Previous,
  Newnode  : Listptr;
begin
  Current := L.Head;
  Previous := null;
```

```
      -- create a node and set the data to element
      Newnode := new Listnode;
      Newnode.Element:= Element;
      -- check if the list is empty.
      if Isempty(L)= False then
         loop
            -- need two separate exits, otherwise there will be
            -- an execption at runtime
            exit when Current = null;
            exit when Current.Element >Element;
            Previous := Current;
            Current := Current.Next;
         end loop;
      end if;
      -- do insertion
      Newnode.Prev:= Previous;
      Newnode.Next := Current;
      if Previous = null then
         -- list is empty
         L.Head := Newnode;
      else
         Previous.Next := Newnode;
         if Current /= null then
            Current.Prev := Newnode;
         end if;
      end if;
   end Insert_In_Order;

end Doubly_Linked_List;
```

108 lines: No errors

Test Program

GNAT 3.15p (20020523) Copyright 1992-2002 Free Software Foundation, Inc.

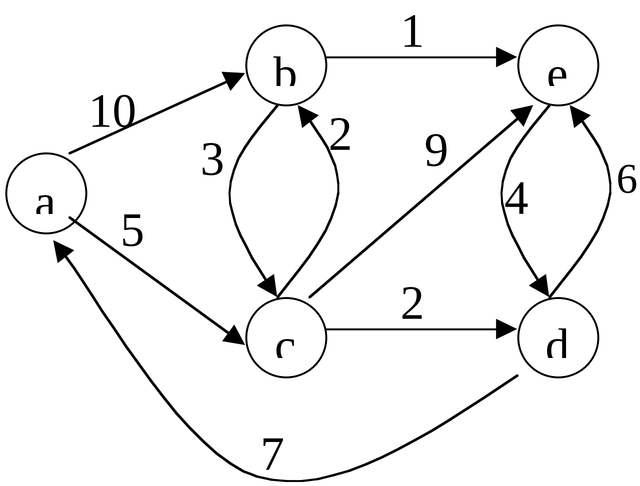
Compiling: c:/docume~1/jayaka~1/mydocu~1/16070/code/doubly_list_test.adb (source file time stamp: 2004-03-31 20:49:02)

```
-------------------------------------------------
-- Program to test the doubly linked list package
-- Programmer: Jayakanth Srinivasan
-- Date Last Modified : Feb 11, 2004
-------------------------------------------------

with Doubly_Linked_List;
use Doubly_Linked_List;

procedure Doubly_List_Test is
   My_List : List;

begin
   -- initialize the list
   Initialize(My_List);

   -- insert 1,3,2,4 into the list and display at each stage
   Insert_In_Order(My_List, 1);
```

```
    Display(My_List);
    Insert_In_Order(My_List, 3);
    Display(My_List);
    Insert_In_Order(My_List, 2);
    Display(My_List);
    Insert_In_Order(My_List, 4);
    Display(My_List);

end Doubly_List_Test;
```

27 lines: No errors

2. What is the Shortest Path through the graph shown below using Dijkstra's algorithm.



Assume node A is the start node.

Show all the steps in the computation of the shortest path.

Initialize

V = {a, b, c, d, e}
E = {(a,b), (a,c), (b,e), (b,c), (c,b), (c,d), (c,e), (d,a), (d,e), (e.d)}
S = {→}
Q = {a, b, c, d, e}

D = [0, ∞, ∞,∞,∞]
Previous = [0, 0, 0, 0, 0]
Start at A

Relax (a,b,10)
Relax (a,c,5)

S = {a}
Q = {b, c, d, e}

D = [0, 10, 5,∞,∞]
Previous = [0,a, a, 0, 0]

Move to C

Relax(c,b,2)
Relax (c,d,2)
Relax(c,e,9)

S = {a,c}
Q = {b, d, e}

D = [0, 7, 5, 7, 14]
Previous = [0,c, a, c, c]

Move to B (you can also move to D)

Relax(b,e,1)
Relax(b,c,3)

S = {a, c, b}
Q = { d, e}

D = [0, 7, 5, 7, 8]
Previous = [0, c, a, c, b]

Move to D

Relax(d,a,7)
Relax(d,e,6)

S = {a, c, b, d}
Q = {e}

D = [0, 7, 5, 7, 8]
Previous = [0, c, a, c, b]

Move to E

Relax(e,d,4)

S = {a, c, b, d, e}
Q = {}

D = [0, 7, 5, 7, 8]
Previous = [0, c, a, c, b]

3. Define the following terms (as applied to graphs):

For a graph G = (V, E). where V is a finite nonempty set of vertices and E is the set of edges,

a. Walk
   A walk is a sequence of vertices (v1, v2, …, vk) in which each adjacent vertex pair is an edge

b. Path
   A path is a walk with no repeated nodes

c. Eulerian Path
   An Eulerian path in a graph is a path that uses each edge precisely once.

d. Cycle
   A cycle is a path that begins and ends with the same vertex

e. Degree of a vertex
   The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex

   In a directed graph, the degree of the vertex is partitioned into **indegree** (number of edges entering a vertex) and **outdegree** (number of edges leaving a vertex). Note that a loop contributes to both the indegree and the outdegree.