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?

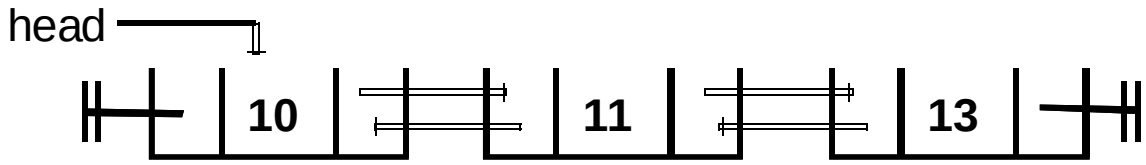


Figure 1. Sample Doubly Linked List

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.

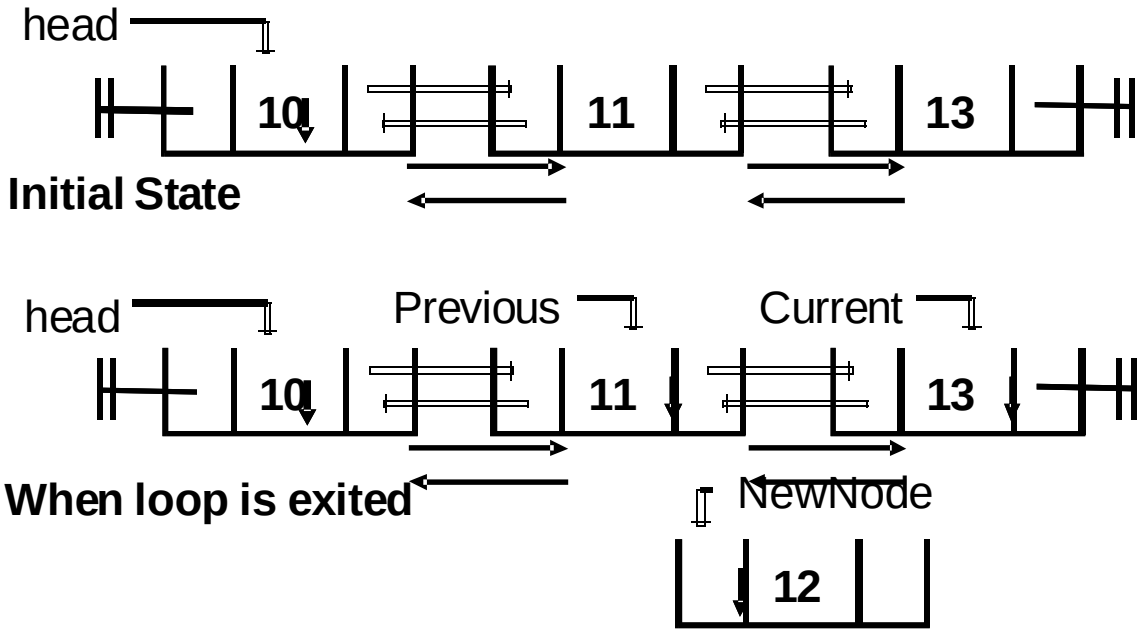
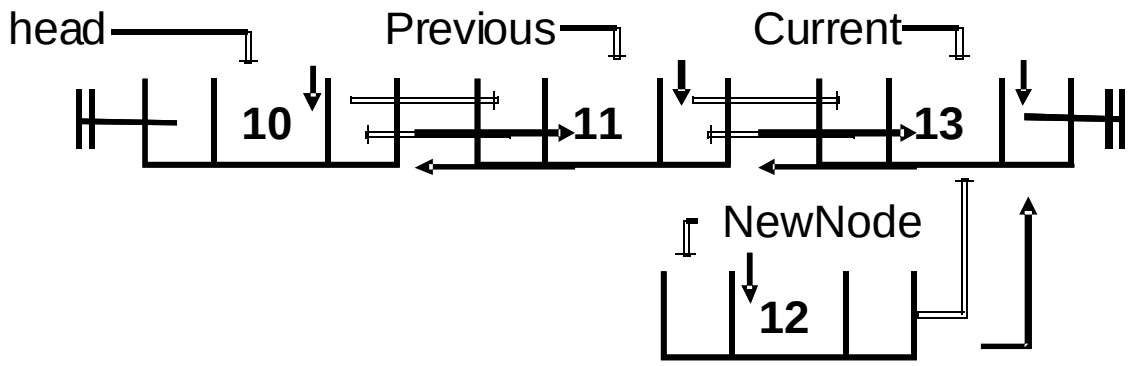
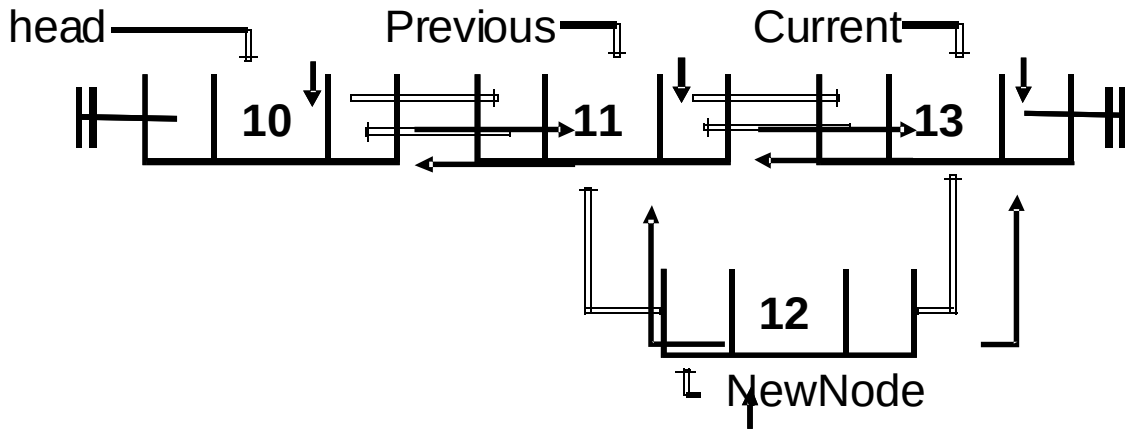


Figure 2. Prior to Insertion

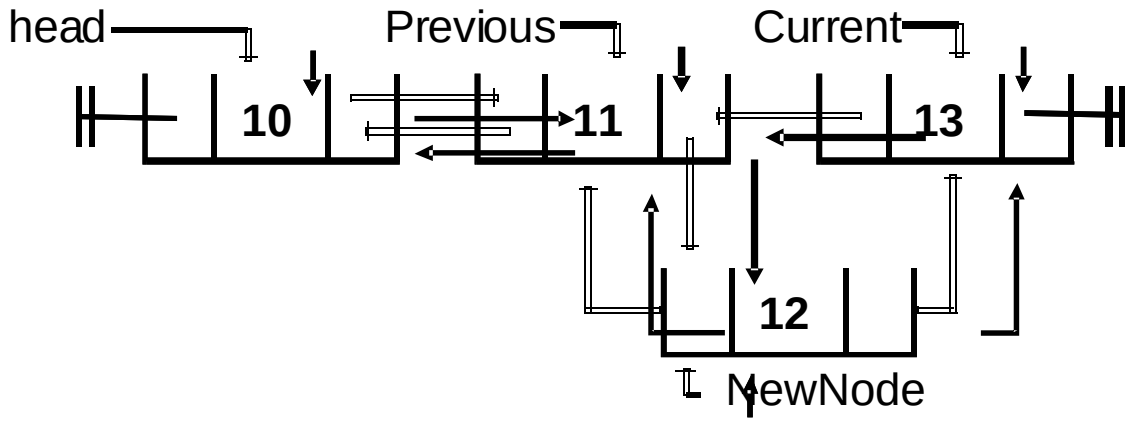
NewNode.Next := Current



NewNode.Prev := Previous



Previous.Next := NewNode



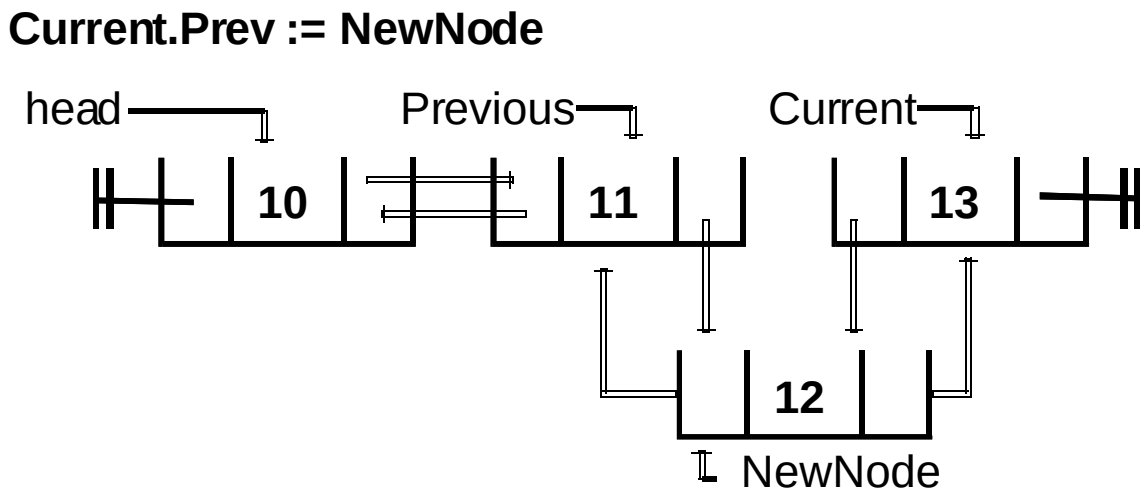


Figure 3. After Insertion Operation

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)

```
 1. ------------------------------------------------------------------------
 2. -- Specification for  doubly-linked lists
 3. -- Specified: Jayakanth Srinivasan
 4. -- Last Modified: February 11, 2004
 5. ------------------------------------------------------------------------
 6.
 7. package Doubly_Linked_List is
 8.
 9.    subtype Elementtype is Integer;
10.
11.    type Listnode;
12.    type Listptr is access Listnode;
13.    type Listnode is
14.       record
15.          Element : Elementtype;
16.          Next    : Listptr;
17.          Prev    : Listptr;     -- this is the change made to singly linked lists
18.       end record;
19.
20.    type List is
21.       record
22.          Head : Listptr;
23.       end record;
24.
25.
26.
27.
```

```
28.    procedure Makeempty (
29.         L : in out List );
30.    -- Pre:    L is defined
31.    -- Post:   L is empty
32.
33.    function Isempty (
34.         L : in     List )
35.      return Boolean;
36.    -- Pre:    L is defined
37.    -- Post:   returns True if L is empty, False otherwise
38.
39.    procedure Display (
40.         L : in     List );
41.    -- Pre:  L may be empty
42.    -- Post: displays the contents of L's Element fields, in the
43.    --   order in which they appear in L
44.
45.    procedure Initialize (
46.         L : in out List );
47.
48.    -- Pre: L may be empty
49.    -- Post: Elements inserted into the list at correct position
50.    procedure Insert_In_Order (
51.         L       : in out List;
52.         Element : in     Elementtype );
53.
54. 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)

```
1. ------------------------------------------------------------------------
2. -- Implementation for  doubly-linked lists
3. -- Programmer: Jayakanth Srinivasan
4. -- Last Modified: Feb 11, 2004
5. ------------------------------------------------------------------------
6.
7. with Ada.Text_Io;
8. with Ada.Integer_Text_Io;
9. with Ada.Unchecked_Deallocation;
10.
11. use Ada.Text_Io;
12. use Ada.Integer_Text_Io;
13.
14. package body Doubly_Linked_List is
15.    -- create an instance of the free procedure
16.    procedure Free is
17.    new Ada.Unchecked_Deallocation(Listnode, Listptr);
18.    -- check if list is empty. List.Head will be null
19.    function Isempty (
20.         L : in     List )
21.      return Boolean is
```

```
22.  begin
23.     if L.Head = null then
24.        return True;
25.     else
26.        return False;
27.     end if;
28.
29.  end Isempty;
30.
31.  -- free all allocated memory at the end of the program
32.  procedure Makeempty (
33.        L : in out List ) is
34.     Temp : Listptr;
35.
36.  begin
37.     loop
38.        exit when Isempty(L);
39.        Temp := L.Head;
40.        L.Head := Temp.Next;
41.        Free(Temp);
42.     end loop;
43.     L.Head := null;
44.  end Makeempty;
45.
46.  -- initialize the list by setting the head pointed to null
47.  procedure Initialize (
48.        L : in out List ) is
49.  begin
50.     L.Head := null;
51.  end Initialize;
52.
53.  -- displays the contents of the list
54.  procedure Display (
55.        L : in     List ) is
56.     Temp : Listptr;
57.  begin
58.     -- set the pointer to the head of the node
59.     Temp:= L.Head;
60.     while Temp /= null loop
61.        Put(Temp.Element);
62.        Put(" , ");
63.        -- move pointer to the next node
64.        Temp :=Temp.Next;
65.     end loop;
66.     New_Line;
67.  end Display;
68.
69.  -- insert elements in ascending order
70.  --  this procedure added :)
71.  procedure Insert_In_Order (
72.        L       : in out List;
73.        Element : in     Elementtype ) is
74.     Current,
75.     Previous,
76.     Newnode  : Listptr;
77.  begin
78.     Current := L.Head;
79.     Previous := null;
```

```
80.      -- create a node and set the data to element
81.      Newnode := new Listnode;
82.      Newnode.Element:= Element;
83.      -- check if the list is empty.
84.      if Isempty(L)= False then
85.        loop
86.          -- need two separate exits, otherwise there will be
87.          -- an execption at runtime
88.          exit when Current = null;
89.          exit when Current.Element >Element;
90.          Previous := Current;
91.          Current := Current.Next;
92.        end loop;
93.      end if;
94.      -- do insertion
95.      Newnode.Prev:= Previous;
96.      Newnode.Next := Current;
97.      if Previous = null then
98.        -- list is empty
99.        L.Head := Newnode;
100.     else
101.       Previous.Next := Newnode;
102.       if Current /= null then
103.         Current.Prev := Newnode;
104.       end if;
105.     end if;
106.   end Insert_In_Order;
107.
108. end Doubly_Linked_List;
```

108 lines: No errors

Test Program

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

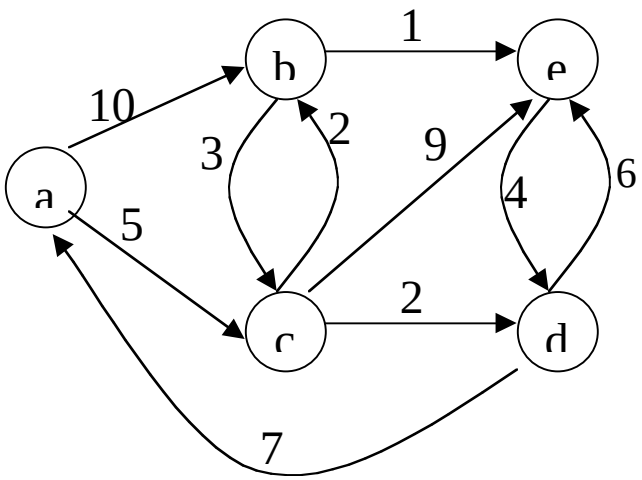
Compiling: c:/docume~1/jayaka~1/mydocu~1/16070/code/doubly_list_test.adb (source file time stamp: 2004-03-31 20:49:02)

```
1. ---------------------------------------------
2. -- Program to test the doubly linked list package
3. -- Programmer: Jayakanth Srinivasan
4. -- Date Last Modified : Feb 11, 2004
5. ---------------------------------------------
6.
7. with Doubly_Linked_List;
8. use Doubly_Linked_List;
9.
10. procedure Doubly_List_Test is
11.    My_List : List;
12.
13. begin
14.    -- initialize the list
15.    Initialize(My_List);
16.
17.    -- insert 1,3,2,4 into the list and display at each stage
18.    Insert_In_Order(My_List, 1);
```

```
19.   Display(My_List);
20.   Insert_In_Order(My_List, 3);
21.   Display(My_List);
22.   Insert_In_Order(My_List, 2);
23.   Display(My_List);
24.   Insert_In_Order(My_List, 4);
25.   Display(My_List);
26.
27. 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.