1. (a) (5 pts.) Identify the tree in Figure 1 that is not a binary search tree (BST) and explain the reason?

(b) (10 pts.) Write a function, **`NODE* search_key(NODE* root, int key)`**, in C language to search the given BST for the node containing the requested key. The structure **`NODE`** is defined to be.

```
typedef struct node
{
   int  key;              /* key value */
   struct node* left;     /* pointer to the left sub-tree*/
   struct node* right;    /* pointer to the right sub-tree*/
}NODE;
```

The first parameter in the function is the pointer to the root node of the given BST and the second parameter is the requested key. The function must return the node containing the requested key. It returns ***`NULL`*** if there is no node containing the requested key.

(c) (15 pts.) Write a function, **`NODE* tree_copy(NODE* root)`**, in C language to make a copy of the given BST. The parameter ***`root`*** is a pointer to the root node. The function returns a pointer to the root of the new copy.

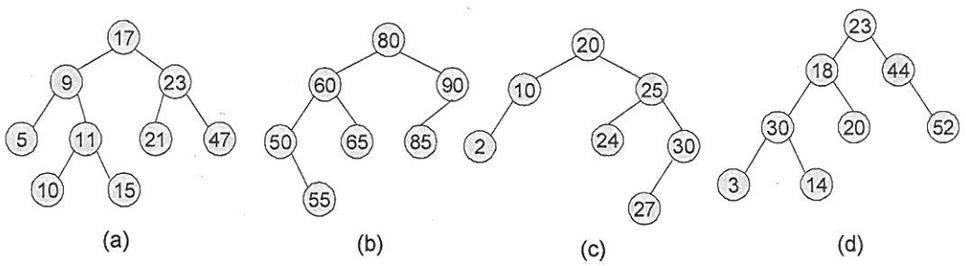


Figure 1

2. Figure 2(a) and Figure 2(b) are examples for a singly linked list and a circular linked list, respectively. The list in Figure 2(c) is the result after performing link inversion on Figure 2(b).

(a) (5 pts.) Define the data structure of the node in the list.

(b) (5 pts.) Write a function in C language or pseudocode to convert a given singly linked list to a circular linked list.

(c) (10 pts.) Write a function in C language or pseudocode to invert the links of a circular list.

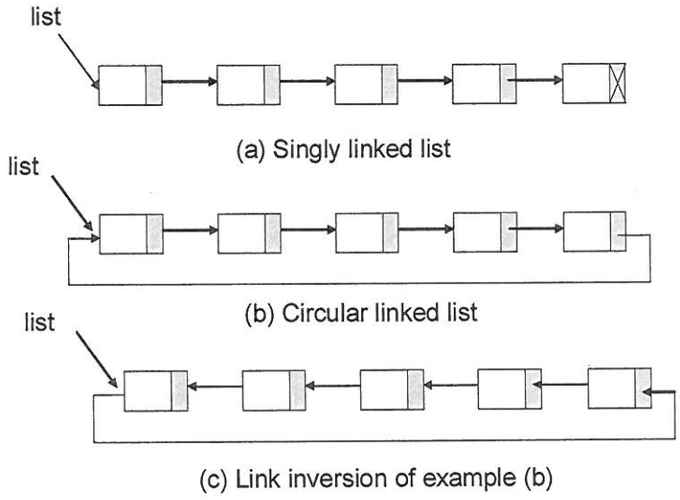
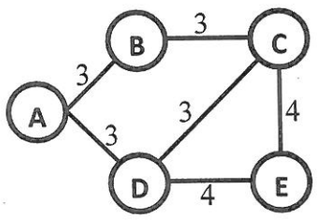


Figure 2.

3. Consider a graph and answer the following.

(a) (10 pts.) Using the following graph, find and draw all minimal spanning trees.



(b) (5 pts.) Define an adjacency list for a graph like above. You may use C or pseudocode. Show the content of your list using the graph above.

(c) (5 pts.) Define an adjacency matrix using C or pseudocode. Show the content of your matrix using the graph above.

(d) (10 pts.) Giving an adjacency list, copy the content of the list to the corresponding adjacency matrix. Using C or pseudocode to implement such function which takes an adjacency list defined in (b), an adjacency matrix defined in (c) as parameters. You can also use more parameters for your function, such as the number of nodes in the graph. You can also limit the maximum number of nodes in the graph.

4. Consider sorting techniques and answer the following.

(a) (10 pts.) Using C or pseudocode to write a function to perform either ***Selection Sort*** or ***Insertion Sort***. Be sure to state which one you are implementing.

(b) (5 pts.) Analyze the above function and show why the complexity is $O(n^2)$. Be sure to show all detail calculation.

(c) (5 pts) Discuss the reasons why sorting techniques with complexity $O(n \log n)$ like ***Quick Sort*** and ***Merge Sort*** may use ***Selection Sort*** or ***Insertion Sort*** in a part of the implementation.