Data Structures

1. struct node{
   
   int data;
   
   node *left;
   
   node *right;

}

The node definition above could be used for the implementation of a

(a) doubly linked list
(b) binary tree
(c) heap
(d) all of the above
(e) none of the above

2. The following sequence of operations is performed on an initially empty stack (represented by the variable s):

s.push(8); s.push(7); s.pop(x); s.push(3); s.push(6); s.pop(x);

s.push(1); s.push(2); s.push(5); s.pop(x); s.pop(x);

Which of the following correctly represents the resultant state of the stack, top to bottom, and the final value of x?

top -> bottom

(a) 1, 3, 8 x=5
(b) 8, 3, 1 x=5
(c) 5, 2, 1 x=6
(d) 1, 2, 5 x=6
(e) none of the above

3. Insert the following integers into a binary search tree: 10, 6, 8, 3, 2

Which diagram correctly represents the resultant binary search tree?

4. The average performance of quicksort is:

(a) $\Theta(n^2)$
(b) $\Theta(n \log n)$
(c) $\Theta(2^n)$
(d) $\Theta(n)$
(e) $\Theta(\log n)$
5. If no duplicate values are stored, the value contained in any node of a binary search tree is:
   (a) smaller than the value of either of its children (if any exist)
   (b) larger than the value of its left child but smaller than the value of its right child (if any exist)
   (c) larger than the value of either of its children (if any exist)
   (d) arbitrary

6. int findsum(node* top)
   {
     if (top == NULL)
       return 0;
     return (______________________________);
   }

The function shown above computes the number of nodes stored in a binary tree whose top node is pointed to by “top.” The blank line should contain which of the following expressions?
   (a) 1 + top->left + top->right;
   (b) top->left + top->right;
   (c) 1 + findsum(top->left) + findsum(top->right);
   (d) top->data + findsum(top->data);
   (e) none of the above

7. Which of the following member functions for a linked list class correctly prints each of the data items contained in the list. Assume that the first item in the list is pointed to by head and the nodes of the list are defined as follows:

struct node{
  int data;
  node *next;
}

(a) void linklist::print(){
    node *temp = head;
    if (temp != NULL) {cout << temp.data << endl; temp = temp.next; }
  }

(b) void linklist::print(){
    node *temp = head;
    while (temp != NULL) {cout << temp.data << endl; temp = temp.next; }
  }

(c) void linklist::print(){
    node *temp = head;
    while (temp != NULL) {temp = temp.next; cout << temp.data << endl; }
  }

(d) void linklist::print(){
    node *temp = head.next;
    while (temp != NULL) {cout << temp.data << endl; temp = temp.next; }
  }

(e) none of the above
8. \[\text{int func(int n)}\]
   \[\text{if (n == 0) return 1;}
   \text{else return func(n-1)*2;}\]

The recursive function given above computes

(a) the \textit{nth} Fibonacci number
(b) \textit{n} factorial
(c) \(2^n\)
(d) \(\log_2 n\)
(e) the sum of the first \textit{n} positive integers

9. The following function returns the minimum value in a binary search tree. Assume the nodes of the tree are stored in a linked list with the given structure. The blank should contain:

```c
struct node{
    float data;
    node *lchild;
    node *rchild;
}

float get_least(node *head){
    float val;
    node *temp = head;
    while (temp != NULL){
        val = temp->data;
        temp = _________;
    }
    return val;
}
```

(a) \text{temp->data}
(b) \text{temp->rchild}
(c) \text{head->rchild}
(d) \text{temp->lchild}
(e) \text{data}

10. The diagram above is a graphical representation of:

(a) a doubly linked list
(b) a circularly linked list
(c) a queue
(d) a heap
(e) a stack