Section 1. **Fundamentals**

1. What is the role of the base case in recursive functions?
   (a) 2
   (b) 4
   (c) 6
2. Write a recursive function **int f(int n)** to return the sum of the integers from 0 to n.
   (a)
   ```
   int Sum(int n) {
   int i;
   if (i == 0)
   return i;
   else return i + Sum(i-1);
   }
   ```
   (b) `int Sum(int n)  return n + Sum(n-1);`
   (c) `int Sum(int n)  return n*(n+1)/2;`

1. Write a recursive function **int f(int m, int n)** to return the sum of the integers from m to n ($m < n$).
2. Write a recursive function **int f(int[] A, int indx)** that counts and returns the number of zeros in the array.
3. Write a recursive function **int SumOfSquares f(int N)** that sums the squares of the first N integers and returns the result.
4. Write a recursive function **int multiply(int i, int j)** that returns the product of two integers, where the multiplication process is defined as a series of additions, eg., 4*7 is 7 added to itself 4 times.
5. Write a recursive function, **int expf(int a, int b)** where $a > 0$ and $b >= 0$. Define the exponential process in terms of multiplication, eg., 125 is 5 multiplied to itself 3 times.

Section 3. **Lists, Stacks, Queues**

1. Discuss the tradeoffs between linked lists and array based structures in terms of storage requirements.
2. Maintaining freelists are a means to minimize overhead associated with linked lists. Explain.
3. Discuss the tradeoffs on using an array vs. a linked list in implementing stacks or queues.
4. Is there any advantage of using doubly linked list (compared to singly linked list) for implementing a queue?
5. Is there any advantage of using doubly linked list (compared to singly linked list) for implementing a stack?
6. Why use a circular array implementation for a queue, as opposed to a regular array?

Section 4. **Binary Trees, Binary Search Trees, Heaps**

1. Distinguish between binary trees and binary search trees.
2. Which method of traversing a tree would result in a sorted list for a binary search tree? Why?
3. How would you find the smallest element in a binary search tree? For a tree with n nodes, what is the complexity(Big Oh) of this operation?
4. Given a preorder traversal of a binary search tree, can you rebuild the tree? Justify in either case.
5. Given an inorder traversal of a binary search tree, can you rebuild the tree? Justify in either case.
6. Given an postorder traversal of a binary search tree, can you rebuild the tree? Justify in either case.
7. How would you find the largest element in a binary search tree? For a tree with n nodes, what is the complexity(Big Oh) of this operation?
8. List one benefit of 2-3 tree over binary search tree.
9. How can the max heap be used for implementing a priority queue?
10. Which data structure is faster for searching: heap or binary search tree? Why?
11. What does it mean for a heap to be complete?
12. Does a heap ever have to be rebalanced? Why?
13. When does a node in a 2-3 tree split?
14. How can splitting a node in a 2-3 tree affect the rest of the tree?

Section 5. **Graphs**

1. What is the definition of path? Of cycle?
2. Adjacency matrix representation is preferred for dense or sparse graphs? Explain.
3. Adjacency list representation is preferred for dense or sparse graphs? Explain.
4. What is a minimum spanning tree, in reference to a weighted graph?
5. Why are shortest path problems important? Name an application.

Section 6. **Hash Tables**

1. Define load factor in a hash table. How does it affect hash table size?
2. What is a perfect hashing function?

Section 7. **General:Comparing Data Structures**

1. If 99.9999% of operation is searching for a specific element, which data structure (out of list, stack, queue, binary tree, binary search tree, heap, graph) would you use? Why?
2. Consider the following applications and relate them to an appropriate data structure:
   - List of cities and flying times between them : _____

- Getting a burger at MacDonalds: _____
- Calling subroutines in a program : _____
- Sending encrypted messages to an ally: _____
- Scheduling jobs on a supercomputer: _____

3. Consider binary search trees vs. heaps(priority queues). If the tree and a heap contained the same set of n keys, which data structure would be faster in writing out the keys in order? Explain why.

4. Consider binary search trees vs. heaps(priority queues). If the tree and a heap contained the same set of n keys, is one faster than the other in writing out all the keys (in no particular order)? Explain.

Answer Key for Exam A

Section 1. **Fundamentals**

1. What is the role of the base case in recursive functions?
   (a) 2
   (b) 4
   (c) 6
2. Write a recursive function **int f(int n)** to return the sum of the integers from 0 to n.
   (a)
   ```
   int Sum(int n) {
   int i;
   if (i == 0)
   return i;
   else return i + Sum(i-1);
   }
   ```
   (b) int Sum(int n) return n + Sum(n-1);
   (c) int Sum(int n) return n*(n+1)/2;

1. Write a recursive function **int f(int m, int n)** to return the sum of the integers from m to n ($m < n$).

   Answer: int Sum(int n) return n + Sum(n-1);

2. Write a recursive function **int f(int[] A, int indx)** that counts and returns the number of zeros in the array.
3. Write a recursive function **int SumOfSquares f(int N)** that sums the squares of the first N integers and returns the result.
4. Write a recursive function **int multiply(int i, int j)** that returns the product of two integers, where the multiplication process is defined as a series of additions, eg., 4*7 is 7 added to itself 4 times.
5. Write a recursive function, **int expf(int a, int b)** where $a > 0$ and $b >= 0$. Define the exponential process in terms of multiplication, eg., 125 is 5 multiplied to itself 3 times.

Section 3. **Lists, Stacks, Queues**

1. Discuss the tradeoffs between linked lists and array based structures in terms of storage requirements.
2. Maintaining freelists are a means to minimize overhead associated with linked lists. Explain.
3. Discuss the tradeoffs on using an array vs. a linked list in implementing stacks or queues.
4. Is there any advantage of using doubly linked list (compared to singly linked list) for implementing a queue?
5. Is there any advantage of using doubly linked list (compared to singly linked list) for implementing a stack?

6. Why use a circular array implementation for a queue, as opposed to a regular array?

Section 4. Binary Trees, Binary Search Trees, Heaps

1. Distinguish between binary trees and binary search trees.
2. Which method of traversing a tree would result in a sorted list for a binary search tree? Why?
3. How would you find the smallest element in a binary search tree? For a tree with n nodes, what is the complexity(Big Oh) of this operation?
4. Given a preorder traversal of a binary search tree, can you rebuild the tree? Justify in either case.
5. Given an inorder traversal of a binary search tree, can you rebuild the tree? Justify in either case.
6. Given an postorder traversal of a binary search tree, can you rebuild the tree? Justify in either case.
7. How would you find the largest element in a binary search tree? For a tree with n nodes, what is the complexity(Big Oh) of this operation?
8. List one benefit of 2-3 tree over binary search tree.
9. How can the max heap be used for implementing a priority queue?
10. Which data structure is faster for searching: heap or binary search tree? Why?
11. What does it mean for a heap to be complete?
12. Does a heap ever have to be rebalanced? Why?
13. When does a node in a 2-3 tree split?
14. How can splitting a node in a 2-3 tree affect the rest of the tree?

Section 5. Graphs

1. What is the definition of path? Of cycle?
2. Adjacency matrix representation is preferred for dense or sparse graphs? Explain.
3. Adjacency list representation is preferred for dense or sparse graphs? Explain.
4. What is a minimum spanning tree, in reference to a weighted graph?
5. Why are shortest path problems important? Name an application.

Section 6. Hash Tables

1. Define load factor in a hash table. How does it affect hash table size?
2. What is a perfect hashing function?

Section 7. General:Comparing Data Structures

1. If 99.9999% of operation is searching for a specific element, which data structure (out of list, stack, queue, binary tree, binary search tree, heap, graph) would you use? Why?
2. Consider the following applications and relate them to an appropriate data structure:
   - List of cities and flying times between them : _____
   - Getting a burger at MacDonalds: _____
   - Calling subroutines in a program : _____
   - Sending encrypted messages to an ally: _____
   - Scheduling jobs on a supercomputer: _____
3. Consider binary search trees vs. heaps(priority queues). If the tree and a heap contained the same set of n keys, which data structure would be faster in writing out the keys in order? Explain why.
4. Consider binary search trees vs. heaps(priority queues). If the tree and a heap contained the same set of n keys, is one faster than the other in writing out all the keys (in no particular order)? Explain.