Q1: Write the differences between arrays and linked lists. Also explain advantages of circular lists.

Differences between Arrays and Linked Lists:

Feature	Arrays	Linked Lists
Storage	Contiguous memory allocation.	Non-contiguous memory allocation.
Size	Fixed at compile-time.	Dynamic; can grow/shrink.
Insertion/Deletion	Expensive (requires shifting).	Efficient (pointers can be updated).
Access Time	O(1) (Direct index access).	O(n) (Sequential traversal).
Memory Usage	No extra memory for pointers.	Extra memory for pointers.
Complexity	Simple to implement.	Slightly complex due to pointers.

Advantages of Circular Lists:

1. Efficient traversal: Traversing from the last node to the first node is done in constant time.
2. Continuous looping: Simplifies applications requiring continuous cycling.
3. No NULL end: Circular lists improve efficiency without a NULL at the end.
4. Memory utilization: Reuse nodes efficiently in constrained environments.

Q2: Write C functions for the following:

i) Search an element in the singly linked list

```
void searchElement(Node* head, int key);
```

ii) Concatenation of two singly linked lists

```
void concatenateLists(Node* head1, Node* head2);
```

Q3: Write C functions for the following:

i) Inserting a node at the beginning of a Doubly Linked List

void insertAtBeginning(Node** head, int data);

ii) Deleting a node at the end of the Doubly Linked List

void deleteAtEnd(Node** head);

Q4: Implement a stack using singly linked list with push & pop functions.

void push(int data);

void pop();

Q5: Write C routines for inserting and deleting an element in a Queue using linked lists.

void enqueue(int data);

void dequeue();

Q6: Write a C function to search for a given element in a Binary Search Tree.

BSTNode* searchBST(BSTNode* root, int key);

Q7: What is a graph? Write the adjacency matrix and adjacency list representation for the given graph.

Adjacency Matrix:

A B C D

A |0 1 1 1

B |1 0 0 1

C |1 0 0 1

D |1 1 1 0

Adjacency List:

A: B -> C -> D

B: A -> D

C: A -> D

D: A -> B -> C

Q8: What is dynamic hashing? Explain techniques with examples.

1. Dynamic Hashing using directories

2. Directory-less dynamic hashing

Q9: Define the leftist tree. Give its declaration in C.

```
struct LeftistNode {
    int data;
    LeftistNode* left;
    LeftistNode* right;
    int npl; // Null Path Length
};
```