Aim:

To check whether a singly linked list is a palindrome.

Algorithm:

1. Use two pointers (slow and fast) to find the middle of the list.
2. Reverse the second half of the linked list.
3. Compare the first half and the reversed second half node by node.
4. If all nodes match, the list is palindrome; else, not.
5. (Optional) Restore the original list structure by reversing the second half again.

Code:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    char data;
    struct Node* next;
};

void append(struct Node** head_ref, char
new_data) {
```

```c
    struct Node* new_node = (struct
Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL) {
        *head_ref = new_node;
        return;
    }

    struct Node* temp = *head_ref;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = new_node;
}

struct Node* reverse(struct Node* head) {
    struct Node* prev = NULL;
    struct Node* current = head;
    struct Node* next = NULL;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}

int isPalindrome(struct Node* head) {
    if (head == NULL || head->next == NULL)
        return 1;

    struct Node *slow = head, *fast = head;
    while (fast != NULL && fast->next != NULL)
{
        slow = slow->next;
        fast = fast->next->next;
    }

    // Reverse second half
    struct Node* second_half = reverse(slow);

    // Compare first half and reversed second
half
    struct Node* first_half = head;
    struct Node* temp_second = second_half;
    while (temp_second != NULL) {
        if (first_half->data !=
temp_second->data)
            return 0; // Not palindrome
        first_half = first_half->next;
        temp_second = temp_second->next;
    }
    // (Optional) Restore the list by reversing
second half back
    reverse(second_half);

    return 1; // Palindrome
}

int main() {
    struct Node* head = NULL;

    // Creating list: r -> a -> d -> a -> r
    append(&head, 'r');
    append(&head, 'a');
    append(&head, 'd');
    append(&head, 'a');
    append(&head, 'r');

    if (isPalindrome(head))
        printf("The linked list is a
palindrome\n");
    else
        printf("The linked list is NOT a
palindrome\n");

    return 0;
}
```

Input:

Linked list: r -> a -> d -> a -> r

Output:

```
The linked list is a palindrome
```

Result:

Palindrome check on SLL completed successfully.