Given a singly linked list, find middle of the linked list.

Keshav Sharma (IIT2018130), Eishaan Singh Rao(IIT2018183)

SEMESTER 1, B.Tech, Department of IT

Indian Institute of Information Technology,Allahabad, India

Abstract:In this paper, we have devised an algorithm to find middle of the singly liked list.

I. INTRODUCTION

Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

Using Structures, we can implement the concept of Linked lists which are linked to one another via internal links to the next node.

Linked list are a data structure which does not require a contiguous memory as in arrays and consist of a data & a pointer to node which points to the next node.

II. ALGORITHM DESCRIPTION

The algorithm used in this program is centred over the following steps:

Step 1:We declare a user defined data type using structure named as Node which has an integer data and a pointer to the next node.

Step 2:Now we declare abstract node named as head as a global variable. It represents head of the linked list.

Step 3:In main function we take input from the user through user defined function named Insert.

Step 4:The control then passes to Print function which is intended to find the middle element and then print it.

Step 5:In the Print function we first transverse alone the complete linked list to find the total number of nodes linked together.

Step 6: Now after we find the total number of nodes. We divide the number by 2 and then traverse the linked list again keeping the count of nodes we have crossed.

Step 7: As we reach the middle node we print the data, at that node and exit the program.

III. ALGORITHM AND ANALYSIS

Algorithm:
Input : Numbers to form a Linked List that is to be operated upon
Output result :The data on the middle node of the linked list obtained from the user.
Method :Linked list, Structure, Pointers, Functions.

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

struct Node{    //User de-
fined data type known as
structure
    int data;
    struct Node* next;  //
Link to the next Node
```

```
};

struct Node* head;  //global variable, i.e., accessible from anywhere in the program

void Insert(int x){
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));  //type casting void* to struct Node*
    temp -> data = x;   //inserting data in the Node temp
    temp -> next = head;
    head = temp;    //since it is an insert from the front
}

void Print(){
    int num=0;
    struct Node* temp = head;
    while(temp != NULL){
        num++;
        temp = temp -> next;
//traversing the list to obtain no. of nodes
    }
    struct Node* temp1 = head;
    for(int i=0; i<(num/2-1); i++){
        temp1 = temp1 -> next;  //traversing again to obtain the middle node
    }
    printf("%d\n", temp1 -> data);
    printf("\n");
}

int main(){
    head = NULL;
    int x;
    for(int i=0; i<5; i++){
        scanf("%d", &x);
        Insert(x);  //taking input for the Linked list
    }
    Print();    //To print the data of middle node
    return 0;
}
```

AlgEnds

CONCLUSION

```
4 7 3 4 5
4

Program ended with exit code: 0
```

From this paper we are able to input a Linked list with the help of structures and traverse through it to find the middle node of that whole list and print the data stored in it.

IV. REFERENCES

This is a completely original work by Keshav Sharma(IIT2018130) and Eishaan Singh Rao(IIT2018183).