IT DS 201 LAB

SUBMITTED BY ADITYA SINGH 2K19/EP/005

Program : Implement Tower of Hanoi problem using Stack.

CODE

```
void tohIterative(int num_of_disks, struct Stack *src,
                  struct Stack *aux, struct Stack *dest){

    int i, total_num_of_moves;
    char s = 'S', d = 'D', a = 'A';

    if (num_of_disks % 2 == 0){
        char temp = d;
        d = a;
        a = temp;
    }
    total_num_of_moves = pow(2, num_of_disks) - 1;

    for (i = num_of_disks; i >= 1; i--)
        push(src, i);

    for (i = 1; i <= total_num_of_moves; i++){
        if (i % 3 == 1)
            moveDisksBetweenTwoPoles(src, dest, s, d);

        else if (i % 3 == 2)
            moveDisksBetweenTwoPoles(src, aux, s, a);

        else if (i % 3 == 0)
            moveDisksBetweenTwoPoles(aux, dest, a, d);
    }
}
```

```
void moveDisk(char fromPeg, char toPeg, int disk){
    cout <<"Move the disk " << disk <<" from "
        << fromPeg << " to "<< toPeg << endl;
}

void moveDisksBetweenTwoPoles(struct Stack *src, struct Stack *dest, char s, char d){
    int pole1TopDisk = pop(src);
    int pole2TopDisk = pop(dest);


    if (pole1TopDisk == INT_MIN){
        push(src, pole2TopDisk);
        moveDisk(d, s, pole2TopDisk);
    }
    else if (pole2TopDisk == INT_MIN){
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }
    else if (pole1TopDisk > pole2TopDisk){
        push(src, pole1TopDisk);
        push(src, pole2TopDisk);
        moveDisk(d, s, pole2TopDisk);
    }
    else{
        push(dest, pole2TopDisk);
        push(dest, pole1TopDisk);
        moveDisk(s, d, pole1TopDisk);
    }
}
```

ALGORITHM

1. Calculate the total number of moves required i.e. "pow(2, n) - 1"
   here n is the number of disks.
2. If the number of disks (i.e. n) is even then interchange destination pole and auxiliary pole.
3. for i = 1 to total number of moves:
   if i%3 == 1: legal movement of top disk between source pole and destination pole
   if i%3 == 2: legal movement top disk between source pole and auxiliary pole
   if i%3 == 0: legal movement top disk between auxiliary pole and destination pole

INPUT/OUTPUT

```c
int main(){

    unsigned num_of_disks = 3;

    struct Stack *src, *dest, *aux;

    src = createStack(num_of_disks);
    aux = createStack(num_of_disks);
    dest = createStack(num_of_disks);

    tohIterative(num_of_disks, src, aux, dest);
    return 0;
}
```

```
Move the disk 1 from S to D
Move the disk 2 from S to A
Move the disk 1 from D to A
Move the disk 3 from S to D
Move the disk 1 from A to S
Move the disk 2 from A to D
Move the disk 1 from S to D
[Finished in 11.4s]
```

END

Program : Write a program to split a given linked list into two sub-list as Front sub-list and Back sub-list, if odd number of the element then add the last element into the front list.

CODE

```
void frontBackSplit(struct Node* source, struct Node** frontRef,
                    struct Node** backRef){
    if (source == NULL || source->next == NULL){
        *frontRef = source;
        *backRef = NULL;
        return;
    }
    struct Node* slow = source;
    struct Node* fast = source->next;

    while (fast != NULL){
        fast = fast->next;
        if (fast != NULL){
            slow = slow->next;
            fast = fast->next;
        }
    }
    *frontRef = source;
    *backRef = slow->next;
    slow->next = NULL;
}
```

```c
struct Node{
    int data;
    struct Node* next;
};

void printList(struct Node* head){
    struct Node* ptr = head;
    while (ptr){
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
    printf("null\n");
}

void push(struct Node** head, int data){
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}
```

ALGORITHM

Fast/Slow Pointer Strategy

1. It uses two pointers to traverse the list.
2. Slow Pointer advances one node simultaneously, while the fast pointer goes two nodes at a time.
3. When the fast pointer reaches the end, the slow pointer will be halfway.
4. Split the list at the right point.

INPUT/OUTPUT

```c
int main(void){

    int keys[] = {6, 3, 4, 8, 2, 9};
    int n = sizeof(keys)/sizeof(keys[0]);

    struct Node* head = NULL;

    for (int i = n-1; i >= 0; i--) {
        push(&head, keys[i]);
    }
    struct Node *a = NULL, *b = NULL;
    frontBackSplit(head, &a, &b);

    printf("Front List: ");
    printList(a);

    printf("Back List: ");
    printList(b);

    return 0;
}
```

```
Front List: 6 -> 3 -> 4 -> 8 -> 2 -> null
Back List: 9 -> 8 -> 6 -> 5 -> null
[Finished in 328ms]
```

END