UCS 1302 Data Structures

Tutorial 1 – SET 1

--Dr. R. Kanchana

1. Given an array of size n and multiple values around which we need to left rotate the array. How to quickly find multiple left rotations? Print the array using O(n) time with O(1) space

Examples:

Input : arr[] = {1, 3, 5, 7, 9}

k1 = 1

k2 = 3

k3 = 4

k4 = 6

Output : 3 5 7 9 1

7 9 1 3 5

9 1 3 5 7

3 5 7 9 1

Input : arr[] = {1, 3, 5, 7, 9}

k1 = 14

Output : 9 1 3 5 7

Solution 1:

**Method 1 (Using temp array)**

Input arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2, n =7

1) Store the first d elements in a temp array

temp[] = [1, 2]

2) Shift rest of the arr[]

arr[] = [3, 4, 5, 6, 7, 6, 7]

3) Store back the d elements

arr[] = [3, 4, 5, 6, 7, 1, 2]

**Time complexity :** O(n)   
**Auxiliary Space :**O(d)

Solution 2:

Method 2: (Without temp array)

// C++ implementation of left rotation of

// an array K number of times

#include <bits/stdc++.h>

using namespace std;

// Function to leftRotate array multiple times

void leftRotate(int arr[], int n, int k)

{

    /\* To get the starting point of rotated array \*/

    int mod = k % n;

    // Prints the rotated array from start position

    for (int i = 0; i < n; i++)

        cout << (arr[(mod + i) % n]) << " ";

    cout << "\n";

}

// Driver Code

int main()

{

    int arr[] = { 1, 3, 5, 7, 9 };

    int n = sizeof(arr) / sizeof(arr[0]);

    int k = 2;

      // Function Call

    leftRotate(arr, n, k);

    k = 3;

      // Function Call

    leftRotate(arr, n, k);

    k = 4;

      // Function Call

    leftRotate(arr, n, k);

    return 0;

}

**Time complexity :** O(n)   
**Auxiliary Space :**O(1)

2. Write a nonrecursive procedure to reverse a singly linked list in O(1) time.

struct node\* reverseList(struct node \*header)

{

struct node \*prev,\*current;

prev=header; current=header->next;

header->next=NULL;

while(current!=NULL)

{

//next=current->next;

current->next=prev;

prev=current;

current=current->next;

}

current=prev;

//return current;

}

3. Reverse the doubly Linked list with O(n) time

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*prev,\*next;

};

struct node\* createList(struct node \*header, int no)

{

struct node \*ptr,\*tail;

struct node \*temp=(struct node\*)malloc(sizeof(struct node));

temp->data=no;

temp->prev=NULL;

temp->next=NULL;

if(header==NULL)

{

temp->next=header;

temp->prev=header;

header=temp;

tail=temp;

}

else

{

//printf("Hello");

struct node \*p1=header;

//printf("%d",p1->data);

p1->prev=temp;

temp->next=p1;

temp->prev=header;

header=temp;

//printf("%d",header->data);

}

return header;

}

void display(struct node \*header)

{

struct node \*ptr=header;

while(ptr!=NULL)

{

printf("\n Data: %d",ptr->data);

ptr=ptr->next;

}

}

int main()

{

printf("Hello World");

struct node \*header;

//header=(struct node\*)malloc(sizeof(struct node));

header=NULL;

header=createList(header,4);

header=createList(header,6);

header=createList(header,14);

createList(header,24);

display(header);

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*