DS-WEEK-4

PRE LABTASK:

PGNO69:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Node {

char name[20];

struct Node \*prev;

struct Node \*next;

};

struct Node \*head = NULL;

// Function to insert a new friend

void insertFriend(char name[20]) {

struct Node \*newFriend = (struct Node\*) malloc(sizeof(struct Node));

strcpy(newFriend->name, name);

newFriend->next = head;

newFriend->prev = NULL;

if (head != NULL)

head->prev = newFriend;

head = newFriend;

}

// Function to delete a friend

void deleteFriend(char name[20]) {

struct Node \*curr = head;

while (curr != NULL) {

if (strcmp(curr->name, name) == 0) {

if (curr->prev != NULL)

curr->prev->next = curr->next;

if (curr->next != NULL)

curr->next->prev = curr->prev;

if (curr == head)

head = curr->next;

free(curr);

break;

}

curr = curr->next;

}

}

// Function to search for a friend

void searchFriend(char name[20]) {

struct Node \*curr = head;

while (curr != NULL) {

if (strcmp(curr->name, name) == 0) {

printf("Friend found: %s\n", curr->name);

break;

}

curr = curr->next;

}

if (curr == NULL)

printf("Friend not found\n");

}

// Function to display all friends

void displayFriends() {

struct Node \*curr = head;

printf("Friends:\n");

while (curr != NULL) {

printf("%s\n", curr->name);

curr = curr->next;

}

}

int main() {

insertFriend("Hari");

insertFriend("Lalitha");

insertFriend("John");

insertFriend("Jane");

displayFriends();

searchFriend("Lalitha");

deleteFriend("John");

displayFriends();

return 0;

}

IN LAB TASK:

Ds Pg no -73:

<https://www.hackerrank.com/challenges/insert-a-node-into-a-sorted-doubly-linked-list/problem?h_r=internal-search/>

DoublyLinkedListNode\* sortedInsert(DoublyLinkedListNode\* head, int data) {

DoublyLinkedListNode \*New = create\_doubly\_linked\_list\_node(data);

if (!head)

{

head = New;

return head;

}

else if (data < (head->data))

{

New->next = head;

head->prev = New;

New->prev = NULL;

head = New;

return head;

}

else

{

DoublyLinkedListNode \*temp = head;

while ( ((temp->next) != NULL) && ((temp->next->data) <= data))

temp = temp->next;

if (temp->next != NULL)

{

DoublyLinkedListNode \*next = temp->next;

next->prev = New;

New->next = next;

}

else

New->next = NULL;

temp->next = New;

New->prev = temp;

}

    return head;

PGNO-74:

<https://www.hackerrank.com/challenges/delete-duplicative-value-nodes-from-a-sorted-linked-list/problem/>

SinglyLinkedListNode\* removeDuplicates(SinglyLinkedListNode\* head) {

struct SinglyLinkedListNode\*temp1=head;

struct SinglyLinkedListNode\*temp=head->next;

while(temp!=NULL)

{

if(head->data==temp->data)

{

temp=temp->next;

head->next=temp;

}

else

{

head=temp;

temp=temp->next;

}

}

return temp1;

}

PGNO 75:

[www.hackerearth.com/problem/algorithm/find-the-middle-of-a-given-linked-list-using-recursion/](http://www.hackerearth.com/problem/algorithm/find-the-middle-of-a-given-linked-list-using-recursion/)

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct node{

int data;

struct node\*next;

};

struct node\*createnode()

{

int data;

scanf("%d",&data);

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

newnode->data=data;

newnode->next=NULL;

return newnode;

}

struct node\*head=NULL;

void insertnode()

{

if(head==NULL)

{

head=createnode();

}

else

{

struct node\*newnode=createnode();

struct node\*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

}

}

int count()

{

int i;

struct node\*temp=head;

for(i=0;temp!=NULL;i++)

{

temp=temp->next;

}

return i;

}

void middle()

{

struct node \*temp1,\*temp2=head;

int cou=count();

int mid=(cou/2);

int i=0;

while(i<mid)

{

temp1=temp2;

temp2=temp2->next;

i++;

}

if(cou%2==1)

{

printf("%d",temp2->data);

}

else

{

printf("%d",temp2->data);

}

}

int main()

{

int n;

scanf("%d",&n);

for(int i=0;i<n;i++)

{

insertnode();

}

    middle();

}

POST LAB TASK-PGNO-76

->Coco game, Krishna, Vasu, Shiva, Ganesh, Sathesh, Naveen and Anand are playing at an University ground. The rally tag is given to Anand. Anand stood and ran around the players and push Krishna as his tag. Imitate it as a circular linked list of Single and Double direction list and display all the players name in the list. Also Find who is the shortest person in the list. Enter the name and height of the Players: Krishna 5.2 Vasu 5.1 Shiva 5.3 Ganesh 5.4 Sathesh 4.9 Naveen 5.5 Anand 5.0

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct player {

char name[10];

float height;

struct player \*next;

};

struct player \*head = NULL;

struct player \*last = NULL;

void insert(char name[10], float height) {

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

strcpy(temp->name, name);

temp->height = height;

temp->next = head;

if (head == NULL) {

head = last = temp;

last->next = head;

} else {

last->next = temp;

last = temp;

last->next = head;

}

}

void display() {

struct player \*temp = head;

printf("Players in the list are: ");

do {

printf("%s ", temp->name);

temp = temp->next;

} while (temp != head);

printf("\n");

}

void findShortest() {

float min = head->height;

char name[10];

strcpy(name, head->name);

struct player \*temp = head->next;

while (temp != head) {

if (temp->height < min) {

min = temp->height;

strcpy(name, temp->name);

}

temp = temp->next;

}

printf("The shortest player is: %s\n", name);

}

int main() {

insert("Krishna", 5.2);

insert("Vasu", 5.1);

insert("Shiva", 5.3);

insert("Ganesh", 5.4);

insert("Sathesh", 4.9);

insert("Naveen", 5.5);

insert("Anand", 5.0);

display();

findShortest();

  return 0;

}

OUTPUT:

Players in the list are: Krishna Vasu Shiva Ganesh Sathesh Naveen Anand

The shortest player is: Sathesh

SKILL PROBLEMS:

1-><https://www.hackerrank.com/challenges/print-the-elements-of-a-linked-list/problem?h_r=internal-search/>

void printLinkedList(SinglyLinkedListNode\* head) {

SinglyLinkedListNode\* temp=head;

while(temp!=NULL)

{

printf("%d\n",temp->data);

temp=temp->next;

}

}

2-> <https://www.codechef.com/problems/REC_05/>

#include<stdio.h>

#include<stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* newNode(int data) {

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

temp->data = data;

temp->next = NULL;

return temp;

}

struct Node\* moveLastNodesToFront(struct Node\* head, int n) {

struct Node\* curr = head, \*prev = NULL;

int len = 0;

while (curr != NULL) {

curr = curr->next;

len++;

}

curr = head;

for (int i = 1; i < len - n; i++)

curr = curr->next;

prev = curr;

curr = curr->next;

prev->next = NULL;

prev = curr;

while (curr->next != NULL)

curr = curr->next;

curr->next = head;

return prev;

}

void printList(struct Node\* head) {

while (head != NULL) {

printf("%d ", head->data);

head = head->next;

}

printf("\n");

}

int main() {

int t, n;

scanf("%d", &t);

while (t--) {

struct Node\* head = NULL, \*curr = NULL;

int x;

scanf("%d", &x);

while (x != -1) {

if (head == NULL) {

head = newNode(x);

curr = head;

} else {

curr->next = newNode(x);

curr = curr->next;

}

scanf("%d", &x);

}

scanf("%d", &n);

head = moveLastNodesToFront(head, n);

printList(head);

}

return 0;

}

3-> <https://www.hackerrank.com/challenges/insert-a-node-at-the-tail-of-a-linked-list/problem?h_r=internal-search/>

SinglyLinkedListNode\* insertNodeAtTail(SinglyLinkedListNode\* head, int data) {

f (head == NULL) {

return create\_singly\_linked\_list\_node(data);

}

SinglyLinkedListNode \*node = head;

while (node->next != NULL) {

node = node->next;

}

node->next = create\_singly\_linked\_list\_node(data);

return head;i

}

4-> <https://www.hackerrank.com/challenges/delete-a-node-from-a-linked-list/problem?h_r=internal-search/>

SinglyLinkedListNode\* deleteNode(SinglyLinkedListNode\* llist, int position) {

if((position) == 0) {

return llist->next;

}

llist->next = deleteNode(llist->next, position-1);

return llist;

}

5-> <https://www.hackerearth.com/problem/algorithm/remove-kth-node/>

#include<stdio.h>

#include<stdlib.h>

struct node

{

int a;

struct node \*next;

};

typedef struct node \* N;

N head;

N getnode(int n)

{

N temp=(N)malloc(sizeof(struct node));

temp->a=n;

temp->next=temp;

return temp;

}

int main()

{

N te,temp;

int div,t,n,k,i,j;

scanf("%d",&t);

for(i=0;i<t;i++)

{

head=NULL;

scanf("%d",&n);

scanf("%d",&k);

for(j=1;j<=n;j++)

{

te=getnode(j);

if(head==NULL)

{

head=te;

}

else

{

temp=head;

while(temp->next!=head)

temp=temp->next;

temp->next=te;

te->next=head;

}

}

if(head->next==head)

{

printf("%d\n",head->a);

head=NULL;

continue;

}

temp=head;

while(head->next!=head)

{

div=k%n;

if(div==0)

{

for(j=1;j<=n-1;j++)

temp=temp->next;

}

for(j=1;j<div;j++)

{

temp=temp->next;

}

te=temp;

te=te->next;

if(te==head)

{

head=head->next;

}

temp->next=te->next;

free(te);

temp=temp->next;

n--;

}

printf("%d\n",head->a);

}

return 0;

}

6-> <https://www.hackerrank.com/challenges/compare-two-linked-lists/problem?h_r=internal-search/>

bool compare\_lists(SinglyLinkedListNode\* head1, SinglyLinkedListNode\* head2) {

int res=1;

while(head1 != NULL || head2 != NULL){

if(head1 == NULL) {res=0; break;}

if(head2 == NULL) {res=0; break;}

if(head1->data != head2->data){res=0;break;}

head1=head1->next;

head2=head2->next;

}

return res;

}