DS WEEK-5

PRE LAB TASK:

PGNO-86:

#include <stdio.h>  
#include <stdlib.h>  
  
struct bogie {  
    int id;  
    struct bogie \*next;  
};  
  
struct bogie \*createBogie(int id) {  
    struct bogie \*newBogie = (struct bogie \*) malloc(sizeof(struct bogie));  
    newBogie->id = id;  
    newBogie->next = NULL;  
    return newBogie;  
}  
  
void insertBogie(struct bogie \*\*head, int id) {  
    struct bogie \*newBogie = createBogie(id);  
    newBogie->next = \*head;  
    \*head = newBogie;  
}  
  
void deleteBogie(struct bogie \*\*head, int id) {  
    struct bogie \*temp = \*head, \*prev = NULL;  
    if (temp != NULL && temp->id == id) {  
        \*head = temp->next;  
        free(temp);  
        return;  
    }  
    while (temp != NULL && temp->id != id) {  
        prev = temp;  
        temp = temp->next;  
    }  
    if (temp == NULL) return;  
    prev->next = temp->next;  
    free(temp);  
}  
  
int searchBogie(struct bogie \*head, int id) {  
    struct bogie \*current = head;  
    while (current != NULL) {  
        if (current->id == id)  
            return 1;  
        current = current->next;  
    }  
    return 0;  
}  
  
void printBogies(struct bogie \*head) {  
    printf("Bogies: ");  
    while (head != NULL) {  
        printf("%d ", head->id);  
        head = head->next;  
    }  
    printf("\n");  
}  
  
int main() {  
    struct bogie \*head = NULL;  
    insertBogie(&head, 1);  
    insertBogie(&head, 2);  
    insertBogie(&head, 3);  
    printBogies(head);  
    deleteBogie(&head, 2);  
    printBogies(head);  
    int id = 3;  
    if (searchBogie(head, id))  
        printf("Bogie with id %d found\n", id);  
    else  
        printf("Bogie with id %d not found\n", id);  
return 0;  
}

IN LAB TASK:

PG NO-90:

void reverseList(int arr[], int n) {

int i, j, temp;

for (i = 0, j = n - 1; i < j; i++, j--) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

    }

}

PGNO-91 :

DoublyLinkedListNode\* reverse(DoublyLinkedListNode\* head) {

struct DoublyLinkedListNode \*prev,\*curr,\*next;

curr=head;

prev=NULL;

while(curr)

{

next=curr->next;

curr->next=prev;

curr->prev=next;

if(next==NULL)break;

prev=curr;curr=next;

}

curr->prev=NULL;return curr;

}

PGNO-92:

#include <stdio.h>

int main(){

int num, i, j=0, k;

scanf("%d", &num);

int \*list = (int\*)malloc(sizeof(int)\*num);

int \*result = (int\*)malloc(sizeof(int)\*num);

for(i = 0; i< num; i++){

scanf("%d", (list+i));

}

for(i=0; i<3; i++){

result[i] = list[i];

}

j = 3;

for(k = 1; k< num/3 && list[k\*3+2] != list[0]; k++){

result[i++] = list[k\*3+2];

}

printf("%d\n", i);

for(j=0; j<i; j++){

printf("%d ", result[j]);

}

return 0;

}

POST LAB TASK-PGNO-94:

->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 SESSION PROBLEMS:

PGNO-96:

#include <stdio.h>

#include<stdlib.h>

int \*stack,top;

int main(void)

{

int t,n,k,num;

scanf("%d",&t);

while(t>=1)

{

top=-1;

scanf("%d%d",&n,&k);

stack=(int\*)calloc(n,sizeof(int));

scanf("%d",&num);

top++;

stack[top]=num;

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

{

scanf("%d",&num);

while(k>=1&&num>stack[top]&&top!=-1)

{

top--;

k--;

}

top++;

stack[top]=num;

}

while(k>=1)

{

top--;

k--;

}

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

printf("%d ",stack[i]);

printf("\n");

t--;

free(stack);

}

}

PGNO-97:

bool has\_cycle(SinglyLinkedListNode\* head) {

SinglyLinkedListNode\* slow = head;

SinglyLinkedListNode\* fast = head;

while(fast!=NULL && fast->next!=NULL){

slow = slow->next;

fast = fast->next->next;

if(slow == fast){

return true;

}

}

return false;

}

PGNO-98:

SinglyLinkedListNode\* reverse(SinglyLinkedListNode\* head) {

if (head == NULL) return NULL;

if (head->next == NULL) {

return head;

} else {

SinglyLinkedListNode \*newHead = reverse(head->next);

head->next->next = head;

head->next = NULL;

return newHead;

}

}

PGNO-99:

<https://www.hackerrank.com/challenges/merge-two-sorted-linked-lists/problem/>

SinglyLinkedListNode\* mergeLists(SinglyLinkedListNode\* head1, SinglyLinkedListNode\* head2) {

if (!head1) return head2;

if (!head2) return head1;

SinglyLinkedListNode\* curr1 = head1;

SinglyLinkedListNode\* curr2 = head2;

SinglyLinkedListNode\* new\_list;

SinglyLinkedListNode\* curr\_new;

if (head1->data < head2->data) {

new\_list = head1;

curr1 = curr1->next;

}

else {

new\_list = head2;

curr2 = curr2->next;

}

curr\_new = new\_list;

while (curr1 && curr2) {

if (curr1->data < curr2->data) {

curr\_new->next = curr1;

curr1 = curr1->next;

}

else {

curr\_new->next = curr2;

curr2 = curr2->next;

}

curr\_new = curr\_new->next;

}

if (curr1 != NULL) {

curr\_new->next = curr1;

}

if (curr2 != NULL) {

curr\_new->next = curr2;

}

return new\_list;

}

PGNO-100:

<https://www.hackerrank.com/challenges/insert-a-node-at-the-head-of-a-linked-list/problem/>

SinglyLinkedListNode\* insertNodeAtHead(SinglyLinkedListNode\* llist, int data) {

SinglyLinkedListNode\* newnode = create\_singly\_linked\_list\_node(data);

newnode->next = llist;

return newnode;

}

PGNO-101:

<https://www.hackerrank.com/challenges/get-the-value-of-the-node-at-a-specific-position-from-the-tail/problem/>

int getNode( SinglyLinkedListNode\* head, int positionFromTail )

{

int total = 0;

SinglyLinkedListNode\* nodePtr = head;

// get the number of the total number of node

while( nodePtr )

{

total++;

nodePtr = nodePtr->next;

}

nodePtr = head;

for( int indx = 0; indx < total-positionFromTail-1; indx++ )

{

nodePtr = nodePtr->next;

}

return nodePtr->data;

}