SINGLY

**INSERTION**

#include <stdio.h>

#include <stdlib.h>

void display();

struct Node {

int data;

struct Node\* next;

};

void display(struct Node\* ptr)

{

while (ptr != NULL) {

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

ptr = ptr->next;

}

}

//INSERTION AT BEGINNING

void insertbeg(struct Node\*\* head, int new\_data){

struct Node \*new;

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

new->data = new\_data;

new->next = \*head;

\*head = new;

}

//END

void insertend(struct Node\* head, int data){

struct Node \*ptr, \*temp;

ptr=head;

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

temp->data = data;

temp->next = NULL;

while(ptr->next != NULL){

ptr=ptr->next;

}

ptr->next = temp;

}

//AT

void insertat(struct Node \*head, int data, int pos){

struct Node \*ptr = head;

struct Node \*new;

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

new->data = data;

new->next = NULL;

pos--;

while (pos!=1){

ptr=ptr->next;

pos--;

}

new->next= ptr->next;

ptr->next = new;

}

int main(){

struct Node \*head = NULL;

insertbeg(&head, 20);

insertend(head, 50);

insertat(head,30,2);

display(head);

return 0;

}

**DELETION**

//DELETION

//end

void delend(struct Node \*head){

if(head==NULL){

printf("EMPTY");

}

else if(head->next == NULL){

free(head);

head=NULL;

}

else{

struct Node \*temp = head;

while(temp->next->next != NULL){

temp = temp->next;

}

free(temp->next);

temp->next = NULL;

}

}

//first

void delfirst(struct Node \*\*head){

struct Node \*temp=(\*head);

(\*head)=(\*head)->next;

free(temp);

}

//deletion at a position

void delat(struct Node \*\*head, int pos){

struct Node \*current = (\*head);

struct Node \*previous = (\*head);

if((\*head)==NULL){

printf("empty");

}

else if(pos==1){

(\*head) = current->next;

free(current);

current=NULL;

}

else{

while(pos!=1){

previous = current;

current = current->next;

pos--;

}

previous->next = current->next;

free(current);

current=NULL;

}

}

**REVERSE**

struct Node\* reverse(struct Node\* head){

struct Node\* nex = NULL;

struct Node\* previous = NULL;

while(head!=NULL){

nex = head->next;

head->next = previous;

previous = head;

head = nex;

}

head =previous;

return head;

}

BST

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*right\_child;

struct node \*left\_child;

};

struct node\* new\_node(int x){

struct node \*temp;

temp = malloc(sizeof(struct node));

temp->data = x;

temp->left\_child = NULL;

temp->right\_child = NULL;

return temp;

}

//insertion

struct node\* insert(struct node \* root, int x){

if (root == NULL)

return new\_node(x);

else if (x > root->data)

root->right\_child = insert(root->right\_child, x);

else

root -> left\_child = insert(root->left\_child, x);

return root;

}

//search

struct node\* search(struct node \* root, int x){

if (root == NULL || root->data == x)

return root;

else if (x > root->data)

return search(root->right\_child, x);

else

return search(root->left\_child, x);

}

//min

struct node\* find\_minimum(struct node \* root) {

if (root == NULL)

return NULL;

else if (root->left\_child != NULL)

return find\_minimum(root->left\_child);

return root;

}

//delete

struct node\* delete(struct node \* root, int x) {

if (root == NULL)

return NULL;

if (x > root->data)

root->right\_child = delete(root->right\_child, x);

else if (x < root->data)

root->left\_child = delete(root->left\_child, x);

else { //root=x

if (root->left\_child == NULL && root->right\_child == NULL){ //NO CHILDREN

free(root);

return NULL;

}

else if (root->left\_child == NULL || root->right\_child == NULL){

struct node \*temp;

if (root->left\_child == NULL)

temp = root->right\_child;

else

temp = root->left\_child;

free(root);

return temp;

}

else {

struct node \*temp = find\_minimum(root->right\_child);

root->data = temp->data;

root->right\_child = delete(root->right\_child, temp->data);

}

}

return root;

}

void inorder(struct node \*root){

if (root != NULL)

{

inorder(root->left\_child);

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

inorder(root->right\_child);

}

}

int main() {

struct node \*root;

root = new\_node(20);

insert(root, 5);

insert(root, 1);

insert(root, 15);

insert(root, 9);

insert(root, 7);

insert(root, 12);

insert(root, 30);

insert(root, 25);

insert(root, 40);

insert(root, 45);

insert(root, 42);

inorder(root);

/\*printf("\n");

root = delete(root, 1);

root = delete(root, 40);

root = delete(root, 45);

root = delete(root, 9);

inorder(root);

printf("\n"); \*/

return 0;

}

// Online C compiler to run C program online

#include <stdio.h>

#include <stdlib.h>

//CIRCULAR

struct node {

int data;

struct node\* next;

};

//creation

struct node\* circular(int data){

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

temp->data = data;

temp->next = temp;

return temp;

}

//insert begi

struct node \*insertbeg(struct node \*tail, int data){

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

newp->data = data;

newp->next=tail->next;

tail->next = newp;

return tail;

}

void print(struct node \*tail){

struct node \*p = tail->next;

do{

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

p=p->next;

}

while(p!=tail->next);

}

int main() {

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

tail = circular(35);

tail = insertbeg(tail, 25);

print(tail);

return 0;

}