**DSA PRACTICAL FILE**

(CodeChef & online GDB as C language IDE is used for all the programs)

Practical File programs from 9 to 12

Name: Hardik Chhabra

ECAM-1

Roll no.: 2020UEA6504

**OUTPUT-9**

**Practical-9**

**Q. Write a program to print the elements of a linked list in reverse order without disturbing the linked list.**

#include<stdio.h>

#include<stdlib.h>

struct LL {

int data;

struct LL \*next;

};

typedef struct LL node;

node \*start=NULL;

node\* getnode() {

node\* newnode;

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

printf("\nEnter data: ");

scanf("%d",&newnode->data);

newnode->next=NULL;

return newnode;

}

int menu() {

int ch;

printf("\n------------------------");

printf("\n1-Create a List");

printf("\n2-Traverse the List from L to R");

printf("\n3-Traverse the List from R to L");

printf("\n4-Exit");

printf("\n------------------------");

printf("\nEnter your choice: ");

scanf("%d",&ch);

return ch;

}

void createLL(int n) {

node \*temp, \*newnode;

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

newnode=getnode();

if(start==NULL) {start=newnode;}

else {

temp=start;

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

temp->next=newnode;

}

}

}

int countnode(node \*ptr) {

int ctr=0;

while(ptr!=NULL) {

ctr++;

ptr=ptr->next;

}

return ctr;

}

void traverse() {

node \*temp;

temp=start;

if(start==NULL) {printf("\nList is Empty!!!");}

else {

printf("\nContents of the list are(from L to R): ");

while(temp!=NULL) {

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

temp=temp->next;

}

}

printf("X");

}

void rev\_traverse(node \*start) {

if(start==NULL) {return;}

else {

rev\_traverse(start->next);

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

}

}

int main(void) {

int ch,n;

while(1) {

ch=menu();

switch(ch) {

case 1: if(start==NULL) {

printf("\nEnter the number of nodes you want: ");

scanf("%d",&n);

createLL(n);

printf("\nList created!!!");

}else {printf("List already exists!!!");}

break;

case 2: traverse();

break;

case 3: printf("\nTraversing from R to L: ");

rev\_traverse(start);

break;

case 4: printf("\nExit!!!");

exit(0);

}

}

return 0;

}

**OUTPUT-10**

**Practical-10**

**Q. Write a program to reverse a linked list.**

#include<stdio.h>

#include<stdlib.h>

struct LL {

int data;

struct LL \*next;

};

typedef struct LL node;

node \*start=NULL;

node\* getnode() {

node\* newnode;

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

printf("\nEnter data: ");

scanf("%d",&newnode->data);

newnode->next=NULL;

return newnode;

}

int menu() {

int ch;

printf("\n1-Create a List");

printf("\n------------------------");

printf("\n2-Reverse a List");

printf("\n3-Traverse the List from L to R");

printf("\n------------------------");

printf("\n4-Exit");

printf("\nEnter your choice: ");

scanf("%d",&ch);

return ch;

}

void createLL(int n) {

node \*temp, \*newnode;

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

newnode=getnode();

if(start==NULL) {start=newnode;}

else {

temp=start;

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

temp->next=newnode;

}

}

}

int countnode(node \*ptr) {

int ctr=0;

while(ptr!=NULL) {

ctr++;

ptr=ptr->next;

}

return ctr;

}

void reverse (){

node \*p, \*q, \*r;

p = q = r = start;

p = p->next->next;

q = q->next;

r->next = NULL;

q->next = r;

while (p != NULL)

{

r = q;

q = p;

p = p->next;

q->next = r;

}

start = q;

}

void traverse(node \*temp) {

if(start==NULL) {printf("\nList is Empty!!!");}

else {

printf("\nContents of the list are(from L to R): ");

while(temp!=NULL) {

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

temp=temp->next;

}

printf("X");

}

}

int main(void) {

int ch,n;

while(1) {

ch=menu();

switch(ch) {

case 1: if(start==NULL) {

printf("\nEnter the number of nodes you want: ");

scanf("%d",&n);

createLL(n);

printf("\nList created!!!");

}else {printf("List already exists!!!");}

break;

case 2: reverse();

break;

case 3: traverse(start);

break;

case 4: printf("\nExit!!!");

exit(0);

}

}

return 0;

}

**OUTPUT-11**

**Practical-11**

**Q. Write a program to add two polynomials using linked lists.**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct link {

float coef;

int expo;

struct link \*next;

};

typedef struct link node;

node\* getnode() {

node \*tmp;

tmp =(node \*) malloc(sizeof(node));

printf("\nEnter Coefficient: ");

fflush(stdin);

scanf("%f",&tmp->coef);

printf("\nEnter Exponent: ");

fflush(stdin);

scanf("%d",&tmp->expo);

tmp->next = NULL;

return tmp;

}

node \* create\_poly (node \*p) {

int ch;

node \*temp,\*newnode;

while(1) {

newnode = getnode();

if(p==NULL) {

p= newnode;

}else {

temp = p;

while(temp->next != NULL)

{temp = temp->next;}

temp->next = newnode;

}

printf ("\n Do you Want polynomial node (1 for y/0 for n): ");

scanf("%d",&ch);

if(ch==0) {break;}

}

return p;

}

void display(node \*p) {

node \*t = p;

while (t != NULL) {

printf("+ %.2f", t->coef);

printf("x^ %d", t->expo);

t = t -> next;

}

}

void add\_poly(node \*p1, node \*p2) {

node \*newnode;

while(1) {

if( p1 == NULL || p2 == NULL )

break;

if(p1->expo == p2->expo) {

printf("+ %.2f X ^%d",p1->coef+p2->coef,p1->expo);

p1 = p1->next; p2 = p2->next;

}else {

if(p1->expo < p2->expo) {

printf("+ %.2fX ^%d",p1->coef,p1->expo);

p1 = p1->next;

}else {

printf(" + %.2fX ^%d",p2->coef,p2->expo);

p2 = p2->next;

}

}

}

while(p1 != NULL) {

printf("+ %.2f X ^%d",p1->coef,p1->expo);

p1 = p1->next;

}

while(p2 != NULL) {

printf("+ %.2f X ^%d",p2->coef,p2->expo);

p2 = p2->next;

}

}

void main() {

node \*poly1 = NULL , \*poly2 = NULL, \*poly3=NULL;

printf("\nEnter First Polynomial(in ascending-order of exponent): ");

poly1 = create\_poly(poly1);

printf("\nEnter Second Polynomial(in ascending-order of exponent): ");

poly2 = create\_poly(poly2);

printf("\n Enter Polynomial 1: ");

display (poly1);

printf("\n Enter Polynomial 2: ");

display (poly2);

printf("\n Resultant Polynomial : ");

add\_poly(poly1, poly2);

display(poly3);

}

**OUTPUT-12**

**Practical-12**

**Q. Write a program to implement doubly-linked list.**

#include <stdio.h>

#include <stdlib.h>

struct dlinklist {

struct dlinklist \*left;

int data;

struct dlinklist \*right;

};

typedef struct dlinklist node;

node \*start = NULL;

node\* getnode() {

node \* newnode;

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

printf("\n Enter data: ");

scanf("%d", &newnode -> data);

newnode -> left = NULL;

newnode -> right = NULL;

return newnode;

}

int countnode(node \*start) {

if(start == NULL) {return 0;}

else {return (1 + countnode(start -> right));}

}

int menu() {

int ch;

printf("\n 1.Create");

printf("\n------------------------------");

printf("\n 8. Traverse the list from Left to Right ");

printf("\n------------------------------");

printf("\n 10.Count the Number of nodes in the list");

printf("\n------------------------------");

printf("\n 11.Exit");

printf("\n\n Enter your choice: ");

scanf("%d", &ch);

return ch;

}

void createlist(int n) {

int i;

node \*newnode;

node \*temp;

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

newnode = getnode();

if(start == NULL){

start = newnode;

}else {

temp = start;

while(temp -> right)

{temp = temp -> right;}

temp -> right = newnode;

newnode -> left = temp;

}

}

}

void traverse() {

node \*temp;

temp = start;

printf("\n The contents of List: ");

if(start == NULL )

{printf("\n Empty List");}

else {

while(temp != NULL) {

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

temp = temp -> right;

}

printf("X");

}

}

int main(void) {

int ch, n;

while(1) {

ch = menu();

switch(ch) {

case 1: printf("\n Enter Number of nodes to create: ");

scanf("%d", &n);

createlist(n);

printf("\n List created..");

break;

case 2: traverse();

break;

case 3: printf("\n Number of nodes: %d", countnode(start));

break;

case 4: printf("Exit!!!");

exit(0);

}

}

return 0;

}