***ASSIGNMENT-7***

***1. WAP to split the Linked List at the middle and merge the second half as the first half.***

#include<stdio.h>

#include<stdlib.h>

struct node{

int info;

struct node \*next;

};

typedef struct node NODE;

NODE \*create(NODE \*head);

NODE \*split(NODE \*head);

void display(NODE \*head);

NODE \*insertlast(NODE \*head,int data);

int main()

{

NODE \*head = NULL;

int ch;

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

printf("1.Create a linked list\n");

printf("2.split the linked list\n");

printf("3.Display the linked list\n");

printf("4.Exit\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

head = create(head);

break;

case 2:

head = split(head);

break;

case 3:

display(head);

break;

case 4:

exit(1);

}

}

return 0;

}

NODE \*create(NODE \*head)

{

if(head!=NULL)

{

NODE \*p=head,\*q;

while(p!=NULL)

{

q=p;

p = p->next;

free(q);

}

}

head=NULL;

int n;

printf("Enter number of nodes in Linked List\n");

scanf("%d",&n);

int i,x;

printf("Enter %d values : ",n);

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

{

scanf("%d",&x);

head = insertlast(head,x);

}

return head;

}

NODE \*insertlast(NODE \*head,int data)

{

NODE \*tmp,\*p = head;

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

tmp->info = data;

tmp->next = NULL;

if(head==NULL)

return tmp;

while(p->next!=NULL)

p = p->next;

p->next = tmp;

return head;

}

NODE \*split(NODE \*head)

{

if(head==NULL || head->next==NULL)

return head;

NODE \*p=head,\*q=head->next,\*r;

while(q->next!=NULL && q->next->next!=NULL)

{

p = p->next;

q = q->next->next;

}

if(q->next!=NULL){

if(q->next->next==NULL)

q = q->next;}

q->next = head;

r = p->next;

p->next = NULL;

return r;

}

void display(NODE \*head)

{

NODE \*p = head;

if(p==NULL)

printf("List is empty\n");

while(p!=NULL)

{

printf("%d\t",p->info);

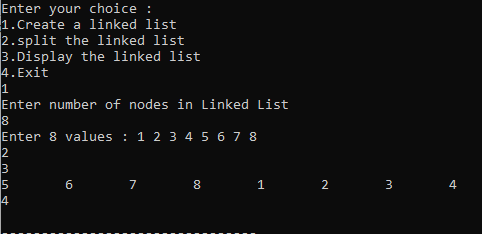
p = p->next;

}

printf("\n");

}

***OUTPUT:***



***2. WAP to implement polynomial addition using Linked List.***

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define SIZE 100

struct node{

int coeff,expo;

struct node \*next;

};

typedef struct node NODE;

NODE \*insertlast(NODE \*head,int x,int y);

NODE \*convert(NODE \*head,char str[]);

NODE \*add(NODE \*head1,NODE \*head2,NODE \*head3);

NODE \*deletelast(NODE \*head);

void display(NODE \*head);

int main()

{

NODE \*head1 = NULL,\*head2 = NULL,\*head3 = NULL;

char str1[SIZE],str2[SIZE];

gets(str1);

gets(str2);

head1 = convert(head1,str1);

head2 = convert(head2,str2);

head3 = add(head1,head2,head3);

printf("Addititon of polynomial is :");

display(head3);

return 0;

}

NODE \*insertlast(NODE \*head,int x,int y)

{

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

tmp->coeff = x;

tmp->expo = y;

tmp->next = NULL;

if(head==NULL)

return tmp;

NODE \*p=head;

while(p->next!=NULL)

p = p->next;

p->next = tmp;

return head;

}

NODE \*convert(NODE \*head,char str[])

{

int l = strlen(str);

int p = 1,q = 0;

int i;

int flag=0;

int sign=0;

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

{

if(str[i]=='-')

sign = 1;

else if(str[i] >='0' && str[i]<='9')

{

p = 0;

while(str[i] >='0' && str[i]<='9' && i<l)

{

p = p\*10 + (str[i]-'0');

i++;

}

i--;

if(sign)

p = -p;

flag=1;

sign=0;

}

else if(str[i]=='x')

{

if(str[i+1]=='^'){

i = i+2;

while(str[i] >='0' && str[i]<='9' && i<l)

{

q = q\*10 + (str[i]-'0');

i++;

}

i--;

}

else

q = 1;

head = insertlast(head,p,q);

p = 1;

q = 0;

flag=0;

}

}

if(flag)

head = insertlast(head,p,q);

return head;

}

NODE \*add(NODE \*head1,NODE \*head2,NODE \*head3)

{

NODE \*p=head1,\*q=head2,\*r=head3;

while(p!=NULL && q!=NULL)

{

if(p->expo == q->expo)

{

head3 = insertlast(head3,((p->coeff)+(q->coeff)),p->expo);

p = p->next;

q = q->next;

}

else if(p->expo < q->expo)

{

head3 = insertlast(head3,q->coeff,q->expo);

q = q->next;

}

else

{

head3 = insertlast(head3,p->coeff,p->expo);

p = p->next;

}

}

if(p!=NULL)

{

while(p!=NULL)

{

head3 = insertlast(head3,p->coeff,p->expo);

p = p->next;

}

}

if(q!=NULL)

{

while(q!=NULL)

{

head3 = insertlast(head3,q->coeff,q->expo);

q = q->next;

}

}

return head3;

}

NODE \*deletelast(NODE \*head)

{

NODE \*p=head,\*q;

while(p->next!=NULL){

q=p;

p = p->next;

}

free(p);

q->next = NULL;

return head;

}

void display(NODE \*head)

{

NODE \*p = head;

int cnt=0;

while(p!=NULL)

{

if(p->expo == 0)

break;

if(p->coeff > 0 && cnt!=0)

printf("+%dx^%d",p->coeff,p->expo);

else

printf("%dx^%d",p->coeff,p->expo);

p = p->next;

cnt++;

}

if(p!=NULL)

{

if(p->coeff > 0)

printf("+%d",p->coeff);

else if(p->coeff < 0)

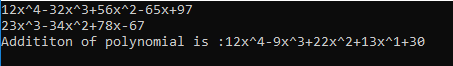
printf("%d",p->coeff);

}

printf("\n");

}

***OUTPUT :***



***3. WAP to add two large numbers using Linked List.***

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<math.h>

#define SIZE 100

struct node{

int info;

struct node \*next;

};

typedef struct node NODE;

NODE \*create(NODE \*head,char str[]);

NODE \*insertstart(NODE \*head,int data);

NODE \*insertlast(NODE \*head,int data);

NODE \*add(NODE \*head1,NODE \*head2,NODE \*head3);

void display(NODE \*head);

int main()

{

char str1[SIZE],str2[SIZE];

gets(str1);

gets(str2);

NODE \*head1=NULL,\*head2=NULL;

head1 = create(head1,str1);

head2 = create(head2,str2);

NODE \*head3 = NULL;

head3= add(head1,head2,head3);

printf("Addition of numbers is : ");

display(head3);

return 0;

}

NODE \*create(NODE \*head,char str[])

{

int i;

int l = strlen(str);

int x=0,cnt=0;

for(i=l-1;i>=0;i--)

{

if(cnt<4)

{

x = x + (str[i]-'0')\*pow(10,cnt);

cnt++;

}

else

{

head = insertlast(head,x);

x = cnt = 0;

i++;

}

}

if(cnt<=4)

head = insertlast(head,x);

return head;

}

NODE \*insertstart(NODE \*head,int data)

{

NODE \*tmp;

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

tmp->info = data;

tmp->next = head;

return tmp;

}

NODE \*insertlast(NODE \*head,int data)

{

NODE \*tmp,\*p = head;

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

tmp->info = data;

tmp->next = NULL;

if(head==NULL)

return tmp;

while(p->next!=NULL)

p = p->next;

p->next = tmp;

return head;

}

NODE \*add(NODE \*head1,NODE \*head2,NODE \*head3)

{

NODE \*p=head1,\*q=head2;

int x=0;

int flag=0;

while(p!=NULL && q!=NULL)

{

x = p->info + q->info;

if(flag==1){

x = x+1;

flag=0;

}

if(x>=10000)

{

flag=1;

x = x-10000;

}

head3 = insertstart(head3,x);

p = p->next;

q = q->next;

}

if(p==NULL && q!=NULL)

{

if(flag==1)

q->info = q->info + 1;

while(q!=NULL)

{

head3 = insertstart(head3,q->info);

q = q->next;

}

}

else if(q==NULL && p!=NULL)

{

if(flag==1)

p->info = p->info + 1;

while(p!=NULL)

{

head3 = insertstart(head3,p->info);

p = p->next;

}

}

else

{

if(flag==1)

head3 = insertstart(head3,flag);

}

return head3;

}

void display(NODE \*head)

{

NODE \*p = head;

if(p==NULL)

printf("List is empty\n");

while(p!=NULL)

{

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

p = p->next;

if(p!=NULL){

if(p->info < 10)

printf("000");

else if(p->info < 100)

printf("00");

else if(p->info < 1000)

printf("0");

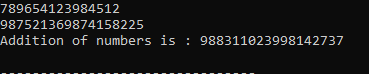
}

}

printf("\n");

}

***OUTPUT :***



***4. WAP to implement Stack using Linked List.***

#include<stdio.h>

#include<stdlib.h>

struct node{

int info;

struct node \*next;

};

typedef struct node NODE;

NODE \*insertlast(NODE \*head,int data);

NODE \*deletelast(NODE \*head);

void display(NODE \*head);

int main()

{

int ch,x;

NODE \*head=NULL;

printf("Enter your choice what you want to do\n");

printf("1.Push an element in Stack\n");

printf("2.Pop an element from Stack\n");

printf("3.Display element of Stack\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter the element you want to push\n");

scanf("%d",&x);

head = insertlast(head,x);

break;

case 2:

head = deletelast(head);

break;

case 3:

display(head);

break;

case 4:

exit(1);

}

}

return 0;

}

NODE \*insertlast(NODE \*head,int data)

{

NODE \*tmp,\*p = head;

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

tmp->info = data;

tmp->next = NULL;

if(head==NULL)

return tmp;

while(p->next!=NULL)

p = p->next;

p->next = tmp;

return head;

}

NODE \*deletelast(NODE \*head)

{

if(head==NULL){

printf("list is empty\n");

return NULL;

}

NODE \*p=head,\*q;

if(head->next==NULL)

{

free(head);

return NULL;

}

while(p->next!=NULL)

{

q=p;

p = p->next;

}

q->next = NULL;

free(p);

return head;

}

void display(NODE \*head)

{

NODE \*p = head;

if(p==NULL)

printf("Stack is empty\n");

while(p!=NULL)

{

printf("%d\t",p->info);

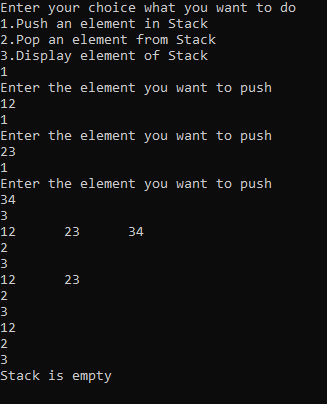
p = p->next;

}

printf("\n");

}

***OUTPUT:***



***5. WAP to implement Queue using Linked List.***

#include<stdio.h>

#include<stdlib.h>

struct node{

int info;

struct node \*next;

};

typedef struct node NODE;

NODE \*insertlast(NODE \*head,int data);

NODE \*deletestart(NODE \*head);

void display(NODE \*head);

int main()

{

int ch,x;

NODE \*head=NULL;

printf("Enter your choice what you want to do\n");

printf("1.Enqueue an element in Queue\n");

printf("2.Dequeue an element from Queue\n");

printf("3.Display element of Queue\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter the element you want to enqueue\n");

scanf("%d",&x);

head = insertlast(head,x);

break;

case 2:

head = deletestart(head);

break;

case 3:

display(head);

break;

case 4:

exit(1);

}

}

return 0;

}

NODE \*insertlast(NODE \*head,int data)

{

NODE \*tmp,\*p = head;

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

tmp->info = data;

tmp->next = NULL;

if(head==NULL)

return tmp;

while(p->next!=NULL)

p = p->next;

p->next = tmp;

return head;

}

NODE \*deletestart(NODE \*head)

{

if(head==NULL)

printf("Queue is empty\n");

else

{

NODE \*tmp = head->next;

free(head);

return tmp;

}

return head;

}

void display(NODE \*head)

{

NODE \*p = head;

if(p==NULL)

printf("Queue is empty\n");

while(p!=NULL)

{

printf("%d\t",p->info);

p = p->next;

}

printf("\n");

}

***OUTPUT :***

