Manoj M Mallya

200905130

Section : C2

Roll no. : 23

LAB – 7 – LINKED LIST CONCEPTS AND APPLICATIONS

Solved Exercises :

1) Implement stack using singly linked list.

Code :

File name: stack_sll_fun.h

```
typedef struct node
{ int info;
struct node *link;
}NODE;
NODE* push(NODE *list,int x)
{ NODE *new,*temp;
new=(NODE*) malloc(sizeof(NODE));
new->link=list;
new->info=x;
return(new);
}
NODE* pop(NODE *list)
{ NODE *prev,*temp;
```

```
if(list==NULL)
{
printf("\nStack Underflow\n");
return(list);
}
temp=list;
 printf(“Deleted element is %d”,temp→info);
 free(temp);
 list = list→link;
return(list);
}
void display(NODE *list)
{
NODE *temp;
printf("\n\nSTACK:");
if(list==NULL)
{
printf(" Stack is empty");
printf("\n\n**********************************************");
return;
}
temp=list;
while(temp!=NULL)
{
printf("%5d",temp->info);
temp=temp->link;
}
```

```
printf(" <- TOP");
printf("\n\n**************************************************");
}
int getchoice()
{
int ch;
printf("**************************************************\n\n");
printf("--------------------Menu--------------------\n");
printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");
printf("Enter your choice:");
scanf("%d",&ch);
return(ch);
}
```

File name: stack_sll.c

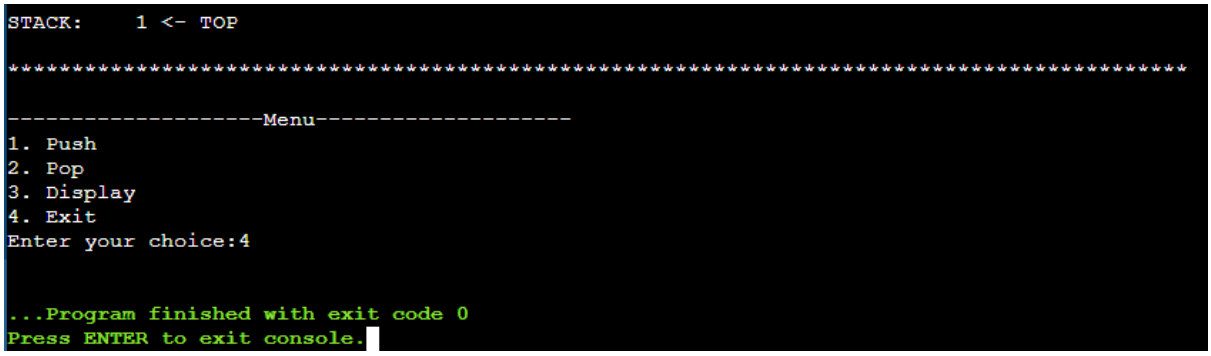
```
#include<stdio.h>
#include "stack_sll_fun.h"
int main()
{
printf("Manoj M Mallya\n200905130\nC2\nRoll no. : 23\n\n\n");
NODE *list;
int x,ch;
list=NULL;
while(1)
{
ch=getchoice();
switch(ch)
```

```
{
case 1: printf("Enter the element to be pushed:");
scanf("%d",&x);
list=push(list,x);
display(list);
break;
case 2: list=pop(list);
display(list);
 break;
case 3: display(list);
getch();
break;
case 4: exit(1);
default: printf("\nInvalid choice");
printf("\n\n**********************************************");
}
}
return 0;
}
```

Output :

```
Manoj M Mallya
200905130
C2
Roll no. : 23


**********************************************

--------------------Menu--------------------
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:1
Enter the element to be pushed:1


STACK:    1 <- TOP

**********************************************************************************************************

--------------------Menu--------------------
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:1
Enter the element to be pushed:2


STACK:    2    1 <- TOP

**********************************************************************************************************
```

```
--------------------Menu--------------------
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:3


STACK:    2    1 <- TOP

**********************************************************************************************************

--------------------Menu--------------------
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:2
Deleted element is 2

STACK:    1 <- TOP

**********************************************************************************************************

--------------------Menu--------------------
1. Push
2. Pop
3. Display
4. Exit
Enter your choice:3


STACK:    1 <- TOP

**********************************************************************************************************
```



2) Given two polynomials, write a program to perform the addition of two polynomials represented using doubly circular linked list with header and display the result.

Code :

File name: poly_add_dll_fun.h

```
struct node
{ int info;
int ex;
struct node *llink;
struct node *rlink;
};
typedef struct node *NODE;
NODE add(NODE head,int n,int e)
{
 NODE temp,last;
temp=(NODE)malloc(sizeof(struct node));
temp->info=n;
temp->ex=e;
last=head->llink;
temp->llink=last;
```

```
last->rlink=temp;
temp->rlink=head;
head->llink=temp;
return head;
}
NODE sum(NODE h1,NODE h2,NODE h3)
{
NODE one,two;
one=h1->rlink;
two=h2->rlink;
while(one!=h1 && two!=h2)
{ if((one->ex)==(two->ex))
{ h3=add(h3,((one->info)+(two->info)),one->ex);
one=one->rlink;
two=two->rlink;
}
else if(one->ex>two->ex)
{ h3=add(h3,one->info,one->ex);
one=one->rlink;
}
else
{ h3=add(h3,two->info,two->ex);
two=two->rlink;
}
}
while(two!=h2)
{ h3=add(h3,two->info,two->ex);
```

```
two=two->rlink;
}
while(one!=h1)
{ h3=add(h3,one->info,one->ex);
one=one->rlink;
}
return h3;
}
void display(NODE head)
{ printf("\ncontents of list are\n");
NODE temp=NULL;
temp=head->rlink;
while(temp!=head)
{ printf("%d %d\t",temp->info,temp->ex);
temp=temp->rlink;
}
}
```

File name: ploy_add_dll.c

```
#include<stdio.h>
#include<stdlib.h>
#include “poly_add_dll_fun.h”
int main()
{
printf("Manoj M Mallya\n200905130\nC2\nRoll no. : 23\n\n\n");
int m,n,e,k;
NODE h1,h2,h3,h4;
```

```
h1=(NODE)malloc(sizeof(struct node));
h2=(NODE)malloc(sizeof(struct node));
h3=(NODE)malloc(sizeof(struct node));
h4=(NODE)malloc(sizeof(struct node));
h1->rlink=h1;
h1->llink=h1;
h2->rlink=h2;
h2->llink=h2;
h3->rlink=h3;
h3->llink=h3;
h4->rlink=h4;
h4->llink=h4;
printf("\nnumber of nodes in list1\n");
scanf("%d",&n);
while(n>0)
{ scanf("%d",&m);
scanf("%d",&e);
h1=add(h1,m,e);
n--;
}
display(h1);
printf("\nnumber of nodes in list2\n");
scanf("%d",&k);
while(k>0)
{ scanf("%d",&m);
scanf("%d",&e);
h2=add(h2,m,e);
```

```
k--;
}
display(h2);
printf("\nthe sum is\n");
h3=sum(h1,h2,h3);
display(h3);
return 1;
}
```

Output :

```
Manoj M Mallya
200905130
C2
Roll no. : 23


number of nodes in list1
3
3 3 3 2 4 1
contents of list are
3 3   3 2   4 1
number of nodes in list2
3
2 3 2 2 1 1
contents of list are
2 3   2 2   1 1
the sum is
contents of list are
5 3   5 2   5 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Questions for Lab7 :

1) Implement a queue using singly linked list without header node.

Code :

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
int data;
```

```
struct node * next;
} * NODE;
NODE enqueue(NODE first, int ele){
NODE temp = (NODE)malloc(sizeof(struct node));
temp->data = ele;
if(first == NULL){
return temp;
}
else{
NODE m = first;
while(m->next != NULL){
m=m->next;
}
m->next = temp;
return first;
}
}
NODE dequeue(NODE first){
if(first == NULL){
printf("\nQueue empty\n");
return NULL;
}
else if(first->next == NULL){
printf("\nDequeue:\t%d\n",first->data);
free(first);
return NULL;
}
```

```
else{
NODE temp = first;
first = first->next;
printf("\nDequeue\t%d\n",temp->data);
free(temp);
return first;
}
}
void display(NODE first){
if(first == NULL){
printf("\nQueue empty\n");
}
else{
NODE temp = first;
while(temp->next != NULL){
printf("%d ",temp->data);
temp=temp->next;
}
printf("%d\n",temp->data);
}
}
int main(){
printf("Manoj M Mallya\n200905130\nC2\nRoll no. : 23\n\n\n");
NODE first = NULL;
int ch,ele;
while(1){
printf("\n1.Enqueue 2.Dequeue 3.Display 4.Exit\nEnter choice : ");
```

```
scanf("%d",&ch);
switch(ch){
case 1: printf("Enter element to Queue: ");
scanf("%d",&ele);
first = enqueue(first,ele);
break;
case 2: first = dequeue(first);
break;
case 3: display(first);
break;
case 4: printf("\nExiting...");
return 0;
}
}
}
```

Output :

```
Manoj M Mallya
200905130
C2
Roll no. : 23



1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 1
Enter element to Queue: 34

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 1
Enter element to Queue: 45

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 1
Enter element to Queue: 78

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 3
34 45 78

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 2

Dequeue 34
```

```
1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 2

Dequeue 34

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 3
45 78

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 4

Exiting...

...Program finished with exit code 0
Press ENTER to exit console.
```

2) Perform UNION and INTERSECTION set operations on singly linked lists with header node.

Code :

#include<stdio.h>

```
#include<stdlib.h>
typedef struct node{
int data;
struct node * next;
} * NODE;
NODE insert(NODE head,int ele){
NODE first = head->next;
NODE temp = (NODE)malloc(sizeof(struct node));
temp->data = ele;
temp->next = NULL;
if(first == NULL){
head->next = temp;
return head;
}
else if(first->next == NULL){
if(first->data > ele){
temp->next = first;
head->next = temp;
return head;
}
else if(first->data < ele){
first->next = temp;
}
else{
printf("\nElement already exists in set.\n");
free(temp);
}
```

```
return head;
}
else{
NODE m = first;
while(m->next != NULL && m->next->data <= ele){
m=m->next;
}
if(m->data != ele){
temp->next = m->next;
m->next = temp;
}
else{
printf("\nElement already exists in the set.\n");
free(temp);
}
return head;
}}
NODE UNION(NODE l1, NODE l2){
NODE uni =(NODE)malloc(sizeof(struct node));
NODE pl1 = l1->next;
NODE pl2 = l2->next;
uni->data = 0;
while(pl1 != NULL && pl2 != NULL){
if(pl1->data < pl2->data){
uni = insert(uni,pl1->data);
pl1 = pl1->next;
}
```

```
else if(pl1->data > pl2->data){
uni = insert(uni,pl2->data);
pl2 = pl2->next;
}
else{
uni = insert(uni,pl1->data);
pl1 = pl1->next;
pl2 = pl2->next;
}}
while(pl1!=NULL){
uni = insert(uni,pl1->data);
pl1 = pl1->next;
}
while(pl2!=NULL){
uni = insert(uni,pl2->data);
pl2 = pl2->next;
}
return uni;
}
NODE INTERSECTION(NODE l1, NODE l2){
NODE inter = (NODE)malloc(sizeof(struct node));
NODE pl1 = l1->next;
inter->data=0;
while(pl1!=NULL){
NODE pl2 = l2->next;
while(pl2!=NULL){
if(pl1->data == pl2->data){
```

```
inter = insert(inter,pl1->data);
break;
}
pl2=pl2->next;
}
pl1=pl1->next;
}
return inter;
}
void display(NODE head){
NODE first = head->next;
if(first == NULL){
printf("\nList empty\n");
}
else{
NODE temp = first;
while(temp->next!=NULL){
printf("%d ",temp->data);
temp=temp->next;
}
printf("%d\n",temp->data);
}}
int main(){
NODE first = (NODE)malloc(sizeof(struct node));
NODE second = (NODE)malloc(sizeof(struct node));
NODE uni = (NODE)malloc(sizeof(struct node));
NODE inter = (NODE)malloc(sizeof(struct node));
```

```
int ch,ele;
first->data = 0;
second->data = 0;
uni->data = 0;
inter->data = 0;
while(1){
printf("\n1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union
6.Intersection 7.Exit\nEnter choice : ");
scanf("%d",&ch);
switch(ch){
case 1: printf("Element : ");
scanf("%d",&ele);
first = insert(first,ele);
break;
case 2: printf("Element : ");
scanf("%d",&ele);
second = insert(second,ele);
break;
case 3: display(first);
break;
case 4: display(second);
break;
case 5: uni = UNION(first,second);
display(uni);
break;
case 6: inter = INTERSECTION(first,second);
display(inter);
break;
```

case 7: printf("\nExiting...\n");

return 0;

}}}

Output :

```
Manoj M Mallya
200905130
C2
Roll no. : 23



1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 1

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 2

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 3

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 3

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 5
```

```
1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 5

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 6

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 3
1 2 3 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 4
3 4 5 6

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 5
1 2 3 4 5 6

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 6
3 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 7

Exiting...


...Program finished with exit code 0
```

****