//DAY-10

/\*//1.Program to save names in SLL, names in the list must be saved in ascending order based on ASCII value

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char name[20];

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

return x;

}

NODE insert(NODE first)

{

NODE temp;

temp=getnode();

temp->link=NULL;

printf("Enter the name:");

scanf("%s",temp->name);

if(first==NULL)

return temp;

NODE cur,prev;

cur=first;

prev=NULL;

if(strcmp(temp->name,cur->name)<0)

{

temp->link=cur;

return temp;

}

while(cur!=NULL && (strcmp(temp->name,cur->name)>0) )

{

prev=cur;

cur=cur->link;

}

prev->link=temp;

temp->link=cur;

return first;

}

void display(NODE first)

{

NODE cur;

cur=first;

if(first==NULL)

{

printf("There are no elements to display");

return ;

}

while(cur!=NULL)

{

printf("%s\n",cur->name);

cur=cur->link;

}

}

int main()

{

int ch;

NODE first=NULL;

for(;;)

{

printf("\nMenu-1.Insert\t2.Display\t3.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:first=insert(first);

break;

case 2:display(first);

break;

case 3:exit(0);

}

}

}

//2.write separate functions to delete a node from SLL based on a)position b)names.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char name[20];

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

return x;

}

void display(NODE first)

{

NODE cur;

cur=first;

if(first==NULL)

{

printf("There are no elements to display");

return ;

}

while(cur!=NULL)

{

printf("%s\t",cur->name);

cur=cur->link;

}

}

NODE insert(NODE first)

{

NODE temp;

temp=getnode();

temp->link=NULL;

printf("Enter the name:");

scanf("%s",temp->name);

if(first==NULL)

return temp;

NODE cur;

cur=first;

while(cur->link!=NULL)

{

cur=cur->link;

}

cur->link=temp;

return first;

}

NODE delete\_info(char item[], NODE first)

{

NODE cur, prev, next,temp;

temp=getnode();

strcpy(temp->name,item);

if(first== NULL)

{

printf("the list is empty\n");

return first;

}

if(strcmp(first->name,temp->name)==0)

{

temp=first;

first=first->link;

printf("Name deleted is %s",temp->name);

free(temp);

return first;

}

prev = NULL;

cur = first;

while(cur != NULL && strcmp(cur->name,temp->name)!=0)

{

prev=cur;

cur=cur->link;

}

if(cur == NULL)

{

printf("Name not found\n");

return first;

}

next=cur->link;

prev->link=next;

printf("Name deleted is %s",cur->name);

free(cur);

free(temp);

return first;

}

NODE delete\_pos(int pos,NODE first)

{

NODE temp,cur,prev,next;

temp=getnode();

if(first==NULL)

{

printf("List is empty\n");

return first;

}

if(pos==1)

{

temp=first;

first=first->link;

printf("Name deleted is %s",temp->name);

free(temp);

return first;

}

int count =1;

prev=NULL;

cur=first;

while(cur!=NULL && count!=pos)

{

prev=cur;

cur=cur->link;

count++;

}

if(cur==NULL)

{

printf("Invalid position\n");

return first;

}

next=cur->link;

prev->link=next;

printf("Name deleted is %s ",cur->name);

free(cur);

return first;

}

int main()

{

int ch;

char item[20];int pos;

NODE first=NULL;

for(;;)

{

printf("\n1.Insert\t2.Delete based on name\t3.Delete based on position\t4.Display\t5.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:first=insert(first);

break;

case 2:

printf("Enter the name to be deleted:");

scanf("%s",item);

first=delete\_info(item,first);

break;

case 3:

printf("Enter the position to be deleted:");

scanf("%d",&pos);

first=delete\_pos(pos,first);

break;

case 4:display(first);

break;

case 5:exit(0);

}

}

}

//3.Write a program to merge two sll and the result must be sorted

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

int empid;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

return x;

}

NODE insert(NODE first)

{

NODE temp;

temp=getnode();

temp->link=NULL;

printf("Enter the empid:");

scanf("%d",&(temp->empid));

if(first==NULL)

return temp;

NODE cur,prev;

cur=first;

prev=NULL;

if(temp->empid<cur->empid)

{

temp->link=cur;

return temp;

}

while(cur!=NULL && (temp->empid>cur->empid))

{

prev=cur;

cur=cur->link;

}

prev->link=temp;

temp->link=cur;

return first;

}

NODE display(NODE first)

{

NODE cur;

cur=first;

if(first==NULL)

{

printf("There are no elements to display");

return NULL ;

}

while(cur!=NULL)

{

printf("%d\t",cur->empid);

cur=cur->link;

}

return first;

}

NODE merge(NODE a, NODE b)

{

NODE last;

if(a==NULL&&b==NULL)

return NULL;

else if(a==NULL)

return b;

else if(b==NULL)

return a;

last=a;

while(last->link!=NULL)

{

last=last->link;

}

last->link=b;

return a;

}

int main()

{

int ch;

NODE a=NULL,b=NULL;

for(;;)

{

printf("\nMenu-1.Insert first LL\t2Insert second LL\t3.Merge\t4.Display\t5.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:a=insert(a);break;

case 2:b=insert(b);break;

case 3:merge(a,b);break;

case 4:display(a);break;

case 5:exit(0);

}

}

}

//4.Program to remove duplicate nodes in SLL

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

return x;

}

void display(NODE first)

{

NODE cur;

cur=first;

if(first==NULL)

{

printf("There are no elements to display");

return ;

}

while(cur!=NULL)

{

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

cur=cur->link;

}

}

NODE insert(NODE first)

{

NODE temp;

temp=getnode();

temp->link=NULL;

printf("Enter the number:");

scanf("%d",&(temp->info));

if(first==NULL)

return temp;

NODE cur;

cur=first;

while(cur->link!=NULL)

{

cur=cur->link;

}

cur->link=temp;

return first;

}

NODE redup(NODE first)

{

if(first==NULL)

{

printf("LL is empty");

return first;

}

NODE temp,cur,prev;

int item;

temp=first;

while(temp!=NULL)

{

item=temp->info;

prev=temp;

cur=temp->link;

while(cur!=NULL)

{

if(item==cur->info)

{

prev->link=cur->link;

free(cur);

cur=prev->link;

}

else

{

prev=cur;

cur=cur->link;

}

}

temp=temp->link;

}

return first;

}

int main()

{

int ch;

NODE first=NULL;

for(;;)

{

printf("\nMenu-1.Insert\t2.RemoveDuplicates\t3.Display\t4.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:first=insert(first);break;

case 2:first=redup(first);break;

case 3:display(first);break;

case 4:exit(0);

}

}

}

\*/