//creation

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \* link;

};

struct node\* head=NULL,\*tail=NULL,\*cur,\*prev,\*next;

void create()

{

int n;

printf("Enter the number of nodes:\n");

scanf("%d",&n);

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

{

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

printf("enter current node data:");

scanf("%d",&(cur->data)); cur->link=NULL;

if(head==NULL)

{

head=tail=cur;

}

else

{

tail->link=cur;

tail=cur;

}

}

}

//insertion at the beginning

void insert\_at\_begin()

{

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

printf("Enter the cur element");

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

cur->link=head;

head=cur;

}

//insertion at end

void insert\_at\_end()

{

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

printf("Enter data");

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

cur->link=NULL;

tail->link=cur;

tail=cur;

}

//insertion at end

void insert\_at\_end()

{

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

printf("Enter data");

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

cur->link=NULL;

tail->link=cur;

tail=cur;

}

//insert at position

void insert\_at\_a\_position()

{

int pos,c=1;

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

printf("Enter the cur data element: \n");

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

printf("Enter the pos to insert:\n");

scanf("%d",&pos);

next=head;

while(c<pos)

{

prev=next;

next=next->link;

c++;

}

prev->link=cur;

cur->link=next;

}

//insert before

void insert\_before()

{

int value;

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

printf("Enter the element to be inserted:\n");

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

printf("Enter data to insert before");

scanf("%d",&value);

next=head;

while(next->data!=value && next!=NULL)

{

prev=next;

next=next->link;

}

prev->link=cur;

cur->link=next;

}

//insert after

void insert\_after()

{

int value;

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

printf("Enter the cur value to be inserted:\n");

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

printf("Enter after which node we need to perform insertion\n");

scanf("%d",&value);

next=head;

while(next->data!=value && next!=NULL)

{

next=next->link;

}

cur->link=next->link;

next->link=cur;

}

void delete\_at\_begin()

{

cur=head;

head=cur->link;

cur->link=NULL;

printf("Deleted element is %d\n",cur->link);

free(cur);

}

//deletion at the ending of list

void delete\_at\_end()

{

cur=head;

while(cur->link!=tail)

{

cur=cur->link;

}

cur->link=NULL;

next=tail;

printf("Deleted element is %d\n",next->data);

free(next);

tail=cur;

}

//deletion before a given node

void delete\_before\_node()

{

int value;

printf("Enter before which node we need to delete");

scanf("%d",&value);

next=head;

while(next->link->data!=value)

{

prev=next;

next=next->link;

}

prev->link=next->link;

next->link=NULL;

printf("Deleted element is %d\n",next->data);

free(next);

}

//deletion after a given node

void delete\_after\_node()

{

int value;

printf("Enter the value after which node we need to delete\n");

scanf("%d",&value);

next=head;

while(next->data!=value)

{

prev=next;

next=next->link;

}

prev=next->link;

next->link=prev->link;

printf("Deleted data is %d\n",prev->data);

prev->link=NULL;

free(prev);

}

//traversal of a single link

void traversal()

{

if(head==NULL)

printf("List is empty");

Else

{

next=head;

}

while(next!=NULL)

{

printf("%d\*->",next->data);

next=next->link;

}

printf("NULL\n");

}

void reverse(struct node\*head)

{

if(head!=NULL)

{

reverse(head->link);

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

}

}

void search\_for\_element()

{

int value,flag=0,c=0;

printf("Enter value to be searched:");

scanf("%d",&value);

next=head;

while(next!=NULL)

{

if(next->data==value)

{

flag=1;

Break;

}

next=next->link;

c++;

}

if(flag==1)

{

printf("Element present in the list at %d position",c+1);

}

else

printf("Element not present in the list");

}

void sorting()

{

struct node\*p1,\*last=NULL;

int i,c,t;

do

{

c=0;

p1=head;

while(p1->link!=last)

{

if(p1->data>p1->link->data)

{

t=p1->data;

p1->data=p1->link->data;

p1->link->data=t;

}

p1=p1->link;

}

last=p1;

}while(c);

}

int main()

{

int ch;

while(1)

{

printf("program for single linked list\n");

printf("1-create\n2-insert at begin\n3-insert at end\n4-insert at position\n5-insert

before");

printf("\n6-insert after\n7-delete at begin\n8-delete at end\n9-delete at

pos\n10-delete before\n");

printf("\n11-delete after\n12-traversal\n13-display in

reverse\n14-search\n15-sort\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:create();

break;

case 2:insert\_at\_begin();

break;

case 3:insert\_at\_end();

break;

case 4:insert\_at\_a\_position();

break;

case 5:insert\_before();

break;

case 6:insert\_after();

break;

case 7:delete\_at\_begin();

break;

case 8:delete\_at\_end();

break;

case 9:delete\_at\_position();

break;

case 10:delete\_before\_node();

break;

case 11:delete\_after\_node();

break;

case 12:traversal();

break;

case 13:reverse(head);

break;

case 14:search\_for\_element();

break;

case 15:sorting();

break;

case 16:exit(0);

}

}

}

OUTPUT:

C:\TDM-GCC-64\dslab>gcc sllfunctions.c -o sllfunctions

C:\TDM-GCC-64\dslab>sllfunctions

program for single linked list

1-create

2-insert at begin

3-insert at end

4-insert at position

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

enter your choice

1

nter the number of nodes:

3

enter current node data:12

enter current node data:32

enter current node data:25

program for single linked list

1-create

2-insert at begin

3-insert at end

4-insert at position

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

enter your choice

12

12\*->32\*->25\*->NULL

program for single linked list

1-create

2-insert at begin

3-insert at end

4-insert at position

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

enter your choice

2

Enter the cur element:11

program for single linked list

1-create

2-insert at begin

3-insert at end

4-insert at position

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

enter your choice 12

11\*->12\*->32\*->25\*->NULL