/\*linked list random deletion and insertion by value\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\*link;

};

struct node\*header;

struct node\*create\_ll(struct node\*);

struct node\*display(struct node\*);

struct node\*insert\_any(struct node\*);

struct node\*delete\_any(struct node\*);

int main()

{

int choice=0;

while(choice!=5)

{

printf("\*\*main menu\*\*\n");

printf("1.create list\n2.display the list\n3.insert at any position\n4.delete from any position\n15.exit\n");

printf("enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:header=create\_ll(header);

break;

case 2:header=display(header);

break;

case 3:header=insert\_any(header);

break;

case 4:delete\_any(header);

break;

case 5:exit(0);

default:

printf("invalid choice\n");

}

}

}

struct node\*create\_ll(struct node\*header)

{

struct node\*new\_node,\*ptr;

int item;

printf("enter -1 to end\n");

printf("enter the data: \n");

scanf("%d",&item);

while(item!=-1)

{

new\_node=(struct node\*)malloc(sizeof(struct node\*));

new\_node->data=item;

if(header==NULL) //list is empty

{

new\_node->link=NULL;

header=new\_node;

}

else

{

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=new\_node;

new\_node->link=NULL;

}

printf("enter the data: \n");

scanf("%d",&item);

}

printf("link list is created\n");

return header;

}

struct node\*display(struct node\*header)

{

printf("the linked list is below\n");

struct node\*ptr;

ptr=header;

while(ptr!=NULL) //list is not empty

{

printf("%d\n",ptr->data);

ptr=ptr->link;

}

return header;

}

struct node\*insert\_any(struct node\*header)

{

struct node\*new\_node,\*ptr;

int val,item;

if(header==NULL)

{

printf("overflow:insertion not possible\n"); //memory bank returns NULL

}

else

{

printf("enter the value after which the node has to be inserted: \n");

scanf("%d",&val);

printf("enter the data to be inserted: \n");

scanf("%d",&item);

new\_node=(struct node\*)malloc(sizeof(struct node\*));

new\_node->data=item;

ptr=header;

while(ptr->data!=val)

{

ptr=ptr->link;

}

new\_node->link=ptr->link;

ptr->link=new\_node;

printf("node inserted at specific position\n");

return header;

}

}

struct node\*delete\_any(struct node\*header)

{

struct node\*ptr,\*ptr1,\*ptr2;

int val;

if(header==NULL)

{

printf("deletion not possible\n"); //list is empty

}

else

{

printf("enter the value after which the node has to be deleted: \n");

scanf("%d",&val);

ptr=header;

ptr1=ptr; //we have to cheak the data of 1st node also

while(ptr1->data!=val)

{

ptr2=ptr1;

ptr1=ptr;

ptr=ptr->link;

}

ptr2->link=ptr1->link;

free(ptr1);

printf("node deleted from specific position\n");

return header;

}

}