DOUBLE LINKED LIST

import java.util.Scanner;

class doubleLL\_node

{

int info;

doubleLL\_node next, prev;

}

public class que3

{

static doubleLL\_node head = null, tail = null; // list empty

static void create()

{

doubleLL\_node p = new doubleLL\_node();

System.out.println("enter info:");

Scanner sc = new Scanner(System.in);

p.info = sc.nextInt();

p.next = p.prev = null;

head = tail = p;

System.out.println("Do you want to continue: y/Y");

char choice = sc.next().charAt(0);

while(choice == 'y' || choice == 'Y')

{

doubleLL\_node q = new doubleLL\_node();

System.out.println("enter info:");

q.info = sc.nextInt();

q.next = null;

p.next = q; // connection bet p & q is established

q.prev = p; // connection bet p & q is established

tail = q; // make the q as tail

p = q;

System.out.println("Do you want to continue: y/Y");

choice = sc.next().charAt(0);

}

print();

}

static void print()

{

if(head == null)

System.out.println("List is empty/underflow");

else

{

doubleLL\_node d = head;

System.out.println("List is:");

while(d!=null)

{

System.out.print(d.info+" ---> ");

d = d.next;

}

}

}

static void reverse\_back\_print()

{

if(tail == null)

System.out.println("List is empty/underflow");

else

{

doubleLL\_node d = tail;

System.out.println("List is:");

while(d!=null)

{

System.out.print(d.info+" ---> ");

d = d.prev;

}

}

}

static void search()

{

Scanner sc = new Scanner(System.in);

if(head==null)

System.out.println("list is empty/underflow");

else

{

doubleLL\_node s = head;

int i = 1, flag=0;

System.out.println("enter element to search:");

int item = sc.nextInt();

while(s!=null)

{

if(item == s.info)

{

System.out.println("element found at location " + i);

flag = 0;

break;

}

else {flag = 1;}

s=s.next;

i++;

}

if(flag==1)

System.out.println("element not found");

}

}

static int count()

{

int c = 0;

if(head == null)

System.out.println("list is empty/underflow");

else

{

doubleLL\_node s = head;

while(s!=null)

{

s=s.next;

c++;

}

}

return c;

}

static void insert()

{

doubleLL\_node p = new doubleLL\_node();

int end = count(); // storing the total no. of nodes

Scanner sc = new Scanner(System.in);

System.out.println("enter new doubleLL\_node info:");

p.info = sc.nextInt();

System.out.println("enter the location to insert:");

int loc = sc.nextInt();

if(loc==1) // insert at front

{

head.prev=p;

p.next = head;

p.prev= null;

head = p;

}

else if(loc==(end+1)) // insert at end

{

tail.next = p;

p.prev = tail;

p.next = null;

tail = p;

}

else if(loc > (end+1))

System.out.println("can't insert after list ends");

else // insert at location other than front & end

{

doubleLL\_node s = head;

for(int i = 1; i< loc-1 && s.next!=null; i++)

{ s = s.next;}

p.next = s.next;

s.next.prev = p;

s.next = p;

p.prev = s;

}

print();

}

static void delete()

{

if(head==null)

System.out.println("list empty/underflow");

else if(head.next == null) // if list contains only a single doubleLL\_node

{

System.out.println("doubleLL\_node deleted: " + head.info);

head = tail = null;

}

else

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter location to delete:");

int loc = sc.nextInt();

int end = count();

if(loc==1) // delete at front

{

System.out.println("doubleLL\_node deleted: " + head.info);

head = head.next;

head.prev = null;

}

else if(loc==end) // delete at end

{

System.out.println("doubleLL\_node deleted: " + tail.info);

tail = tail.prev;

tail.next = null;

}

else // delete at any position

{

doubleLL\_node s = head;

doubleLL\_node q = new doubleLL\_node();

for(int i = 1; i<loc && s.next!=null; i++)

{

q = s;

s = s.next;

}

q.next = s.next;

s.next.prev = q;

System.out.println("doubleLL\_node deleted: "+ s.info);

}

}

print();

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

while(true)

{

System.out.println("\n\*\*\*\*MENU\*\*\*\*\*");

System.out.println("0:Exit");

System.out.println("1:Creation");

System.out.println("2:Insert at any position");

System.out.println("3:Delete at any position");

System.out.println("4:Reverse/Back\_display the list");

System.out.println("5:Search an element");

System.out.println("6:Count the list");

System.out.println("ENTER CHOICE FROM MENU:");

int choice=sc.nextInt();

switch(choice)

{

case 0: System.exit(0);

case 1: create(); break;

case 2: insert(); break;

case 3: delete(); break;

case 4: reverse\_back\_print(); break;

case 5: search(); break;

case 6: int c = count();

System.out.println("Total number of nodes = " + c);

break;

default:

System.out.println("Wrong choice");

}

}

}

}

SINGLE LINKED LIST

import java.util.Scanner;

class singleLL\_node

{

int info;

singleLL\_node next;

}

public class que4

{

static singleLL\_node head = null; // list empty

static void create()

{

singleLL\_node p = new singleLL\_node();

Scanner sc = new Scanner(System.in);

System.out.println("enter info:");

p.info = sc.nextInt();

p.next = null;

head = p; // p is the head now...the value of p is transferred to head

System.out.println("Do you want to continue: y/Y");

char choice = sc.next().charAt(0);

while(choice == 'y' || choice == 'Y')

{

singleLL\_node q = new singleLL\_node();

System.out.println("enter info:");

q.info = sc.nextInt();

q.next = null;

p.next = q; // connection bet p & q is established

p = q;

System.out.println("Do you want to continue: y/Y");

choice = sc.next().charAt(0);

}

print();

}

static void insert()

{

singleLL\_node p = new singleLL\_node();

Scanner sc = new Scanner(System.in);

System.out.println("enter new singleLL\_node info to insert:");

p.info = sc.nextInt();

int end = count(); // finding end of the list

singleLL\_node s = head;

System.out.println("enter the location for insertion:");

int loc = sc.nextInt();

if (loc == 1) // insert at beginning

{

p.next=head;

head=p;

}

else if(loc==(end+1)) // insert at end

{

while(s.next!=null)

{

s=s.next;

}

s.next=p;

p.next=null;

}

else if(loc > (end+1))

System.out.println("can't insert after list ends");

else // otherwise insert at the location other than front & end

{

for(int i=1;i<loc-1 && s.next!=null; i++)

{

s=s.next;

}

p.next= s.next;

s.next=p;

}

print();

}

static void delete()

{

if(head == null)

System.out.println("list empty/underflow");

else if(head.next==null) // if list contains only a single singleLL\_node

{

System.out.println("Deleting: "+ head.info);

head=null;

}

else

{

Scanner sc = new Scanner(System.in);

singleLL\_node s = head;

singleLL\_node q= new singleLL\_node();

System.out.println("enter the location to delete:");

int loc = sc.nextInt();

if(loc == 1) // delete from beginning

{

System.out.println("Deleting: " + head.info);

head = head.next;

}

else // delete at any position

{

int end = count(); // finding end of the list

for(int i =1; i<loc && s.next!=null; i++)

{

q=s;

s=s.next;

}

System.out.println("Deleting: " + s.info);

q.next = s.next;

if(loc==end)

q.next = null;

}

}

print();

}

static void print()

{

singleLL\_node d = head;

if( d == null)

System.out.println("List empty/underflow");

else

{

System.out.println("List is:");

while(d!=null)

{

System.out.print(d.info+" ---> ");

d = d.next;

}

}

}

static void search()

{

Scanner sc = new Scanner(System.in);

singleLL\_node s = head;

int i = 1, flag=0;

if(s==null)

System.out.println("list empty/underflow");

else

{

System.out.println("enter element to search:");

int item = sc.nextInt();

while(s!=null)

{

if(item == s.info)

{

System.out.println("element found at location " + i);

flag = 0;

break;

}

else {flag = 1;}

s=s.next;

i++;

}

if(flag==1)

System.out.println("element not found");

}

}

static int count()

{

int c = 0;

singleLL\_node s = head;

if(s == null)

System.out.println("list empty/underflow");

else

{

while(s!=null)

{

s=s.next;

c++;

}

}

return c;

}

static void sort()

{

singleLL\_node s = head;

singleLL\_node q = new singleLL\_node();

if(s==null)

System.out.println("List empty..can't sort");

else

{

System.out.println("sorting done");

while(s!=null)

{

q = s.next;

while(q!=null)

{

if(s.info > q.info)

{

int x = s.info;

s.info = q.info;

q.info = x;

}

q=q.next;

}

s=s.next;

}

}

print();

}

static void reverse()

{

singleLL\_node s =head;

singleLL\_node r = null;

singleLL\_node q = new singleLL\_node();

if(s==null)

System.out.println("List empty..can't reverse");

else

{ System.out.println("Reversal done");

while(s!=null)

{

q = s.next;

s.next = r;

r = s;

s = q;

}

head = r;

}

print();

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

while(true)

{

System.out.println("\n\*\*\*\*MENU\*\*\*\*\*");

System.out.println("0:Exit");

System.out.println("1:Creation");

System.out.println("2:Insert at any position");

System.out.println("3:Delete at any position");

System.out.println("4:Search an element");

System.out.println("5:Count the list");

System.out.println("6:Reverse the list");

System.out.println("7:Sort the list");

System.out.println("ENTER CHOICE FROM MENU:");

int choice=sc.nextInt();

switch(choice)

{

case 0: System.exit(0);

case 1: create(); break;

case 2: insert(); break;

case 3: delete(); break;

case 4: search(); break;

case 5: int c = count();

System.out.println("Total number of nodes = "+ c );

break;

case 6: reverse(); break;

case 7: sort(); break;

default:

System.out.println("Wrong choice");

}

}

}

}