1…………………………………………..Binary search………………………………………………………………………………..

/\* Here we are implementing binary search function which returns the position

upon successful finding otherwise returns -1.

\*/

public class BinarySearchADT // Here ADT stands for Abstract data type (ie., class)

{

public static void main(String args[])

{

int a[] = {3,5,7,12,17,23,34,45,56,67,77,89,94};

int key,position;

key = Integer.parseInt(args[0]); // search key from command line

position = binarySearch(a,a.length,key);

if(position == -1)

System.out.println("Sorry...No key found...");

else

System.out.println("Key found at "+position+ "location...");

}

public static int binarySearch(int b[],int n,int key)

{

int lb, ub, mid;

lb = 0;

ub = n-1;

while(lb<=ub) // fail means (lb>ub)

{

mid = (lb+ub)/2;

if(b[mid] == key)

return mid;

else

if(b[mid]<key)

lb = mid+1;

else

ub = mid-1;

}

return -1;

}

}

2…………………………………………….fibnocci search………………………………………………………………….

public class FibSearch

{

public static void main(String args[])

{

int a[]={1,3,15,36,45,67,78,91};

int key=8;

int position=fibsearch(a,8,key);

if(position==-1)

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

else

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

}

public static int fibsearch(int a[],int n,long key)

{

int fib[]={0,1,1,2,3,5,8,13,21,34,55,89,144};

int initpos=0,position,k;

if (n==-1)

return -1;

k=0;

while(fib[k]<n)

k++;

while(k>0)

{

position=initpos+fib[--k];

if((position>=n)||key<a[position])

{

continue;

}

else

{

if(a[position]<n)

{

initpos=position+1;

k--;

}

else

return position;

}

}

return -1;

}

}

3……………………………………………………..selection sort……………………………………….

public class FibSearch

{

public static void main(String args[])

{

int a[]={1,3,15,36,45,67,78,91};

int key=8;

int position=fibsearch(a,8,key);

if(position==-1)

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

else

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

}

public static int fibsearch(int a[],int n,long key)

{

int fib[]={0,1,1,2,3,5,8,13,21,34,55,89,144};

int initpos=0,position,k;

if (n==-1)

return -1;

k=0;

while(fib[k]<n)

k++;

while(k>0)

{

position=initpos+fib[--k];

if((position>=n)||key<a[position])

{

continue;

}

else

{

if(a[position]<n)

{

initpos=position+1;

k--;

}

else

return position;

}

}

return -1;

}

}

4…………………………insertion sort………………………………………………….

import java.util.Scanner;

public class InsertionSortADT

{

public static void main(String args[])

{

int a[]; // array reference

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size");

int n = sc.nextInt();

a = new int[n+1]; // dynamic memory allocation in java for array

System.out.println("Enter the array elements");

for(int i=1;i<=n;i++)

a[i] = sc.nextInt();

System.out.println("\n Before sorting...");

display(a,n);

insertionSort(a,n); // calling sorting function

System.out.println("\n After sorting...");

display(a,n);

}

public static void insertionSort(int a[], int n)

{

int i, j, temp;

for(i=1;i<=n;i++)

{

temp = a[i];

j = i-1; // j refers to previous positions

while((j>0) && (temp < a[j]))

{

a[j+1] = a[j]; // a[i-1+1] = a[i-1]

j--;

} // shifting is finished

a[j+1] = temp; // Placing the temp back in the empty position

}

}

public static void display(int a[],int n)

{

for(int i=1;i<=n;i++)

System.out.print(a[i]+" ");

}

}

5…………………….quick sort……………………………….

import java.util.Scanner;

public class QuickSort

{

public static void main (String args[])

{

Scanner sc=new Scanner(System.in);

int i,n,a[];

System.out.println("enter the size");

n=sc.nextInt();

a=new int[n];

System.out.println("enter the elements");

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

a[i]=sc.nextInt();

System.out.println("\nbefore sorting");

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

System.out.print(a[i]+" ");

quickSort(a,0,n-1);

System.out.println("\nafter sorting");

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

System.out.print(a[i]+" ");

}

public static void quickSort(int a[], int left, int right)

{

int pivot, l,u,temp;

l= left;

u = right;

pivot = left; // pivot or key starts always from leftmost item

while(left!=right)

{

while((a[right] >= a[pivot] ) && (left!=right)) // R to L

right--;

if(left!=right) // swap

{

temp = a[pivot];

a[pivot] = a[right];

a[right] = temp;

pivot = right; // change the pos of pivot

}

while(( a[left]<=a[pivot] ) && (left!=right))// L to R

left++;

if(left!=right) // swap

{

temp = a[pivot];

a[pivot] = a[left];

a[left] = temp;

pivot = left; // change the pos of pivot

}

}

if(l<pivot) // apply quicksort if left sublist exists

quickSort(a,l,pivot-1);

if(pivot<u) //apply quicksort if right sublist exists

quickSort(a,pivot+1,u);

}

}

6…………………………………….radix sort………………………………………..

import java.util.\*;

public class RadixSortADT

{

public static void main(String args[])

{

int a[];

int n,i;

Scanner sc = new Scanner(System.in);

System.out.println("Enter no of elements");

n = sc.nextInt();

a = new int[n]; // memory allocation for the array 'a'.

System.out.println("Enter the elements");

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

a[i] = sc.nextInt();

System.out.println("\n Before sorting....");

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

System.out.print(a[i]+" ");

radixSort(a,n);

System.out.println("\n After sorting....");

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

System.out.print(a[i]+" ");

}

public static void radixSort(int a[],int n)

{

int big, count,digit,div;

int bucket[][]; // 10 buckets

int b[]; // bucket count-no of elements in each bucket

int i,j,k,p;

b = new int[10]; // to hold no of elements in each bucket

bucket = new int[10][15]; // allocate memory for all 10 buckets

big = a[0];

for(i=1;i<n;i++) // to find the biggest number in the list

{

if(big<a[i])

big = a[i];

}

count=0;

while(big>0) // finding no of digits in the biggest number

{

big=big/10;

count++;

}

div = 1;

for(p=1;p<=count;p++) // repeated for number times based on the digitcount

{

for(i=0;i<10;i++) // initialize all bucket counts to 0

b[i] = 0;

for(i=0;i<n;i++) // distribution

{

digit = (a[i]/div)%10;

bucket[digit][b[digit]] = a[i]; // Placing the item in the bucket

b[digit]++; // incrementing the respective bucket count

}

i=0;

for(k=0;k<10;k++) // gather all the elements from all the buckets

{

for(j=0;j<b[k];j++)

a[i++] = bucket[k][j];

}

div = div \* 10;

}

}

}

7……………………………merge sort…………………

import java.util.Scanner;

public class MergeSortADT{

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] a;

int i;

System.out.print("Enter the no.of elements of an array : ");

int n = input.nextInt();

a = new int[n];

System.out.println("Enter array elements : ");

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

a[i] = input.nextInt();

}

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

for(int j : a)

System.out.print(j+" ");

System.out.println();

merge(a,0,n-1);

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

for(int j : a)

System.out.print(j+" ");

}

public static void merge(int[] a,int lb,int ub){

if(lb<ub){

int mid=(lb+ub)/2;

merge(a,lb,mid);

merge(a,mid+1,ub);

mergeSort(a,lb,mid,ub);

}

}

public static void mergeSort(int[] a,int lb,int mid,int ub){

int i = lb;

int j = mid+1;

int k=lb;

int[] b = new int[20];

while(i<=mid && j<=ub){

if(a[i]>=a[j]){

b[k] =a[j];

j++;

k++;

}

else{

b[k] = a[i];

i++;

k++;

}

}

while(j<=ub){

b[k]=a[j];

k++;j++;

}

while(i<=mid){

b[k]=a[i];

k++;i++;

}

for(int l=lb;l<=ub;l++)

a[l] = b[l];

}

}

8…………………………stack using array……………………

class Stack

{

int stk[];

int top;

final int MAX = 6;

public Stack()// Constructor initializes Stack & top

{

stk = new int[MAX];

top = 0;

}

public void push(int e)

{

if(isFull())

{

System.out.println("Stack is Overflow ..!");

return;

}

stk[++top] = e;

System.out.println("Item "+e+" pushed...");

}

public int pop()

{

if(isEmpty())

{

System.out.println("Stack is Underflow..!");

return -1;

}

int k = stk[top--];

return k;

}

public boolean isFull()

{

return (top==MAX-1)? true : false ;

}

public boolean isEmpty()

{

return (top == 0)? true : false ;

}

public int peek()

{

return stk[top]; // returns topmost element in the stack

}

public void viewStack()

{

if(isEmpty())

{

System.out.println("No Data in Stack...!");

return;

}

System.out.println("\n The Content of Stack is...\n");

for(int i = top; i>0;i--)

System.out.print(stk[i]+" ");

}

}

public class StackADT

{

public static void main(String args[])

{

Stack obj = new Stack();

obj.push(77);

obj.push(57);

obj.push(47);

obj.push(67);

obj.push(12);

obj.push(80);

obj.viewStack();

System.out.println("Topmost element = "+obj.peek());

System.out.println("Popped..."+obj.pop());

System.out.println("Popped..."+obj.pop());

System.out.println("Popped..."+obj.pop());

System.out.println("Topmost element = "+obj.peek());

obj.push(100);

obj.viewStack();

}

}

9……………………………….infix to postfix……………………………………..

import java.util.Scanner;

class ConversionADT{

char stk[];

int top;

final int MAX = 10;

public ConversionADT(){

stk= new char[MAX];

top = 0;

}

public String infixToPostfix(String infix){

String postfix=" ";

char ch;

for(int i=0;i<infix.length();i++){

ch = infix.charAt(i);

if(Character.isLetter(ch))

postfix += ch;

else

if(ch=='(')

stk[++top] = ch;

else

if(ch == ')'){

while(stk[top]!='(')

postfix += stk[top--];

top--;

}

else {

while( priority(stk[top]) >= priority(ch) )

postfix += stk[top--];

stk[++top] = ch;

}

}

while(top>0)

postfix += stk[top--];

return postfix;

}

public int priority(char ch){

if(ch=='^')

return 5;

else

if((ch=='\*') || (ch=='/') || (ch=='%'))

return 4;

else

if((ch=='+') || (ch=='-'))

return 3;

else

if(ch=='=')

return 1;

else

return -1;

}

}

public class MyConversionADT{

public static void main(String args[]){

String infix,postfix;

Scanner sc = new Scanner(System.in);

System.out.println("Enter any infix notation...");

infix = sc.next(); // input as string

ConversionADT c = new ConversionADT();

postfix = c.infixToPostfix(infix);

System.out.println("The given infix notation..."+infix);

System.out.println("The given Postfix notation..."+postfix);

}

}

10……………………………….post fix eval…………………………….

/\* Evaluation of postfix expression \*/

import java.util.Scanner;

class EvaluateExpression

{

double stk[];

int top;

final int MAX = 10;

Scanner obj;

public EvaluateExpression()

{

stk = new double[MAX];

top = 0;

obj = new Scanner(System.in);

}

public double operation(double p, double q, char op)

{

switch(op)

{

case '^': return Math.pow(p,q);

case '\*': return p\*q;

case '/': return p/q;

case '+': return p+q;

case '-': return p-q;

case '%': return p%q;

}

return 0;

}

public double evalPostfix(String post)

{

double x,y,z;

char ch;

for(int i=0;i<post.length();i++)

{

ch = post.charAt(i);// accepting each letter from postfix

if(Character.isDigit(ch))

stk[++top] = (double)(ch - '0');//Ex:'5'-'0' ==> 53-48=5

else

if(Character.isLetter(ch))

{

System.out.println("Enter value for "+ch+" : ");

int val = obj.nextInt();

stk[++top] = val;

}

else

{

y = stk[top--];

x = stk[top--];

z = operation(x,y,ch);

stk[++top]= z;

}

}

return stk[top]; // final result

}

11……………………………………..queue…………………………………….

class QueueADT

{

String queue[];

final int MAX=5;

int front, rear;

public QueueADT() // initializing the queue

{

queue = new String[MAX];

front = rear = 0;

}

public void enqueue(String e) // to insert one element at a time

{

if(isQueueFull())

{

System.out.println("Queue is Overflow..!");

return;

}

queue[rear++] = e;

System.out.println(e+" Inserted successfully....");

}

public void dequeue() // to remove one element at a time.

{

if(isQueueEmpty())

{

System.out.println("Queue is Underflow...!");

return;

}

String v = queue[front++];

System.out.println("\nElement removed is..."+v);

}

public boolean isQueueEmpty()

{

return (front==rear)?true:false;

}

public boolean isQueueFull()

{

return (rear==MAX)?true:false;

}

public void viewQueue() // to show the content of the queue

{

if(isQueueEmpty())

{

System.out.println("No data found...!");

return;

}

System.out.println("\nThe content is...");

for(int i=front;i!=rear;i++)

System.out.print(queue[i]+" ");

}

}

public class MyQueueADT

{

public static void main(String args[])

{

QueueADT q = new QueueADT();

String k [] = {"Ramu","Ravi","Kiran","Asha","Vasu","Vinay"};

for(int i=0;i<k.length;i++)

q.enqueue(k[i]);

q.viewQueue();

q.dequeue();

q.dequeue();

q.viewQueue();

q.dequeue();q.dequeue();q.dequeue();q.dequeue();

q.viewQueue();

}

}

12………………………single linked list……………………………

/\* Single linked list creation & other operations \*/

import java.util.Scanner;

class Student

{

int stno;

String name;

String branch;

int year;

public Student(int no, String nm,String br,int y)

{

stno = no; name = nm;

branch = br; year = y;

}

public void get()

{

System.out.println(stno+" "+name+" "+branch+" "+year);

}

}

class Node

{

Student data;

Node next;

public Node(Student nm)

{

data = nm;

next = null;

}

}

class LinkedList

{

Node head; // head node means starting node

Scanner sc;

public LinkedList()

{

head = null;

sc = new Scanner(System.in);

}

void createList() // completely creates a linked list

{

while(true)

{

System.out.println("\nEnter Student name...\n");

String k = sc.next();

insert(k);

System.out.println("Wish to insert one more Student?");

String choice = sc.next();

if(choice.equals("no"))

return;

}

}

void insert(String data) // adds one node at a time to the linked list

{

Node temp, t;

temp = new Node(data); // creation of a new node

if(head==null)

head = temp;

else

{

t = head;

while(t.next!=null) // traverse or move to the last node

t = t.next;

t.next = temp;

}

}

void viewLinkedList()

{

Node temp = head;

while(temp!=null)

{

System.out.print(temp.data+"->");

temp = temp.next;

}

System.out.print("null");

}

public int count()

{

int c=0;

Node temp = head;

while(temp!=null)

{

c++;

temp = temp.next;

}

return c;

}

public void reverseDisplay()

{

String stk[] = new String[10];

int top=0;

Node temp = head;

while(temp!=null)

{

stk[++top] = temp.data;

temp = temp.next;

}

for(int i=top;i>=0;i--)

System.out.print(stk[i]+" ");

}

}

public class LinkedListADT

{

public static void main(String args[])

{

LinkedList obj = new LinkedList();

obj.createList();

obj.viewLinkedList();

System.out.println("\nNo of nodes = "+obj.count());

obj.reverseDisplay();

}

}

13………………………………..double linked list………………………………..

import java.util.Scanner;

class MyNode

{

MyNode prev;

int data;

MyNode next;

}

class MyLinkedList

{

Scanner s=new Scanner(System.in);

MyNode head, last;

MyLinkedList()

{

head=null;

last=null;

}

void create(int n)

{

MyNode temp;

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

{

temp=new MyNode();

System.out.println("enter data");

temp.data=s.nextInt();

temp.next=null;

if(head==null)

{

head=last=temp;

head.prev=null;

}

else

{

last.next=temp;

temp.prev=last;

last=temp;

}

}

}

void insertB()

{

MyNode temp =new MyNode();

System.out.println("enter data");

temp.data=s.nextInt();

temp.prev=null;

temp.next=head;

head.prev=temp;

head=temp;

}

void insertE()

{

MyNode temp=new MyNode();

MyNode temp1;

System.out.println("enter data");

temp.data=s.nextInt();

temp.next=null;

for(temp1=head;temp1.next!=null;temp1=temp1.next)

temp.prev=temp1;

temp1.next=temp;

last=temp;

}

void insertP(int p)

{

MyNode temp=new MyNode();

MyNode temp1;

int i;

System.out.println("enter data");

temp.data=s.nextInt();

for(i=0,temp1=head;i<p-1;temp1=temp1.next,i++)

temp.next=temp1.next;

temp.prev=temp1;

temp1.next.prev=temp;

temp1.next=temp;

}

void deleteB()

{

MyNode temp;

temp=head;

head=temp.next;

head.prev=null;

temp=null;

}

void deleteE()

{

MyNode temp, temp1;

for(temp=head;temp.next!=null;temp=temp.next)

temp1=temp.prev;

temp1.next=null;

temp=null;

last=temp1;

}

void deleteP(int p)

{

MyNode temp,temp1;

int i;

for(temp=head, i=0;i<p-1;i++,temp=temp.next)

temp1=temp.next;

temp.next=temp1.next;

temp1.next.prev=temp;

temp1=null;

}

void displayF()

{

MyNode temp;

for(temp=head;temp!=null;temp=temp.next)

System.out.print(temp.data+"->");

System.out.println("null");

}

void displayB()

{

MyNode temp;

for(temp=last;temp!=null;temp=temp.prev)

System.out.print(temp.data+"->");

System.out.println("null");

}

}

class DoubleLinkedListDemo

{

public static void main(String[] args)

{

Scanner s=new Scanner(System.in);

MyLinkedList l=new MyLinkedList();

System.out.println("enter the size of the list");

int ch;

l.create(s.nextInt());

while(true)

{

System.out.println("enter your choice");

ch=s.nextInt();

switch(ch)

{

case 1: l.insertB();

break;

case 2: l.insertE();

break;

case 3:System.out.println("enter the position");

l.insertP(s.nextInt());

break;

case 4:l.deleteB();

break;

case 5:l.deleteE();

break;

case 6: System.out.println("emter the position");

l.deleteP(s.nextInt());

break;

case 7:l.displayF();

break;

case 8:l.displayB();

break;

default:return;

}}}}

14……………………………..round robin……………………………..

function roundrobin(p,a,b,n){

var seq=[]

var res\_b=[];

var res\_a=[];

for (var x = 0; x <b.length; x++) {

res\_b[x] = b[x];

res\_a[x] = a[x];

}

var t=0;

var w=[];

var comp=[];

while(true){

var flag=true;

for(var i=0;i<p.length;i++){

if (res\_a[i]<=t){

if(res\_a[i]<=n){

if(res\_b[i]>0){

flag=false;

if (res\_b[i] > n) {

t = t + n;

res\_b[i] = res\_b[i] - n;

res\_a[i] = res\_a[i] + n;

seq.push(p[i]);

}

else {

t = t + res\_b[i];

comp[i] = t - a[i];

w[i] = t - b[i] - a[i];

res\_b[i] = 0;

seq.push(p[i]);

}

}

}

else if (res\_a[i] > n) {

for (var j = 0; j < p.length; j++) {

if (res\_a[j] < res\_a[i]) {

if (res\_b[j] > 0) {

flag = false;

if (res\_b[j] > n) {

t = t + n;

res\_b[j] = res\_b[j] - n;

res\_a[j] = res\_a[j] + n;

seq.push(p[j]);

}

else {

t = t + res\_b[j];

comp[j] = t - a[j];

w[j] = t - b[j] - a[j];

res\_b[j] = 0;

seq.push(p[j]);

}

}

}

}

if (res\_b[i] > 0) {

flag = false;

if (res\_b[i] > n) {

t = t + n;

res\_b[i] = res\_b[i] - n;

res\_a[i] = res\_a[i] + n;

seq.push(p[i]);

}

else {

t = t + res\_b[i];

comp[i] = t - a[i];

w[i] = t - b[i] - a[i];

res\_b[i] = 0;

seq.push(p[i]);

}

}

}

}

else if (res\_a[i] > t) {

t++;

i--;

}

}

if (flag) {

break;

}

}

document.getElementById("demo1").innerHTML = seq;

document.getElementById("demo2").innerHTML = w;

document.getElementById("demo3").innerHTML = comp;

}