**LAMPIRAN**

**TUGAS 1 MODUL I**

Program *Remove Duplicate*

1. *Source Code*

|  |
| --- |
| class duplicates  {  Node head;  class Node  {  int data;  Node next;  Node(int d) {data = d; next = null; }  }    void removeDuplicates()  {  Node curr = head;    while (curr != null) {  Node temp = curr;  while(temp!=null && temp.data==curr.data) {  temp = temp.next;  }  curr.next = temp;  curr = curr.next;  }  }    public void push(int new\_data)  {  Node new\_node = new Node(new\_data);  new\_node.next = head;  head = new\_node;  }    void printList()  {  Node temp = head;  while (temp != null)  {  System.out.print(temp.data+" ");  temp = temp.next;  }  System.out.println();  }  public static void main(String args[])  {  duplicates llist = new duplicates();  llist.push(20);  llist.push(13);  llist.push(13);  llist.push(12);  llist.push(12);  llist.push(11);  llist.push(11);  llist.push(11);  llist.push(11);    System.out.println("List sebelum duplikat dihapus: ");  llist.printList();    llist.removeDuplicates();    System.out.println("List sesudah duplikat dihapus: ");  llist.printList();  }  } |

1. Hasil program

|  |
| --- |
|  |

Program *Overlap*

1. *Source Code*

|  |
| --- |
| import java.util.Scanner;  public class nomor2 {  int x=0;  class node{  int data;  node next;  node prev;  public node(int data){  this.data=data;  this.next=null;  this.prev=null;}}  node head;  node tail;  public void addLast(int data){  node tmp = new node(data);  x+=1;  if(head==null){  head=tail=tmp;  tail.next=head;  }  else{  tail.next=tmp;  tmp.prev=tail;  tail=tail.next;  tail.next=head;} }  public void print(){  if(head==null){  System.out.print("Empty");  }  else{  node tmp=head;  do{  System.out.print(""+tmp.data+""+" ");  tmp=tmp.next;  }while(tmp!=head);} }  public void geser(int k){  if(head==null){  System.out.print("Empty");  }  else{  node tmp=head;  node tmp1=head;  for(int i=0;i<k;i++){  if(tmp!=null){  tmp=tmp.next;  tmp1=tmp1.next;  }else{  tmp=head;  tmp1=head;  }  }  do{  System.out.print("{"+tmp.data+"}");  tmp=tmp.next;  }while(tmp!=tmp1);  }  }  public static void main(String[] args) {  nomor2 c = new nomor2();  Scanner input = new Scanner(System.in);  c.addLast(3);  c.addLast(2);  c.addLast(4);  c.addLast(5);  c.addLast(6);  c.addLast(1);  c.addLast(7);  c.addLast(9);  System.out.print("Linked list awal: ");  c.print();  System.out.println();  System.out.print("Masukkan nilai K: ");  int k = input.nextInt();  System.out.print("Linked list akhir: ");  c.geser(k);  }  } |

1. Hasil program

|  |
| --- |
|  |

Program *Merge Linked List*

1. *Source Code*

|  |
| --- |
| class nomorTiga {  static class Node {  int data;  Node next;  };  static Node newNode(int key)  {  Node temp = new Node();  temp.data = key;  temp.next = null;  return temp;  }  static void printList(Node node)  {  while (node != null) {  System.out.printf("%d ", node.data);  node = node.next;  }  }    static Node merge(Node h1, Node h2)  {  if (h1 == null)  return h2;  if (h2 == null)  return h1;  if (h1.data < h2.data) {  h1.next = merge(h1.next, h2);  return h1;  }  else {  h2.next = merge(h1, h2.next);  return h2;  }  }  public static void main(String args[])  {  Node head1 = newNode(2);  head1.next = newNode(5);  head1.next.next = newNode(8);    Node head2 = newNode(0);  head2.next = newNode(1);  head2.next.next = newNode(4);  head2.next.next.next = newNode(9);  Node mergedhead = merge(head1, head2);  printList(mergedhead); } } |

1. Hasil program

|  |
| --- |
|  |

**TUGAS 1 MODUL II**

Membuat *Double Linked List* Dengan Banyak *Method*.

1. *Source Code*

|  |
| --- |
| import java.util.Random;  public class tugasmodul2 {  int x=0;  node head, tail, prev;  int size;  class node{  int data;  node next;  node prev;  public node(int data){  this.data=data;  this.next=null;  this.prev=null;  }  }  public boolean isEmpty(){  return x==0;  }  public void addFirst(int data){  node newnode = new node (data);  if(isEmpty()){  head=tail=newnode;  head.prev=tail;  }  else{  head.prev=newnode;  newnode.next=head;  head=newnode;  //head.prev=tail;  }x++;  }  public void addLast(int data){  node newnode = new node (data);  if(isEmpty()){  head=tail=newnode;  tail.next=head;  }  else{  tail.next=newnode;  newnode.prev=tail;  tail=newnode;  //tail.next=head;  }x++;  }  public boolean addAfter(int data, int newdata) {    node current;  current = head;  while (current.data!= data) {  current=current.next;  if(current==null){  return false;  }  }  node bantu=new node (newdata);  if(current==tail){  bantu.next=null;  tail=bantu;  }  else{  bantu.next=current.next;  current.next.prev=bantu;  }  bantu.prev=current;  current.next=bantu;  return true;  }  public void removeFirst(int data){  head=head.next;  head.prev=tail;  }  public void removeLast(int data){  tail=tail.prev;  tail.next=head;  }  public void ganjil(){  node current=head;  if(isEmpty()){  System.out.println("Linked List kosong");  }  while(current!=null){  if((current.data%2)==1){  System.out.print(current.data+" ");  }current=current.next;  }  }  public void genap(){  node current=head;  if(isEmpty()){  System.out.println("Linked List kosong");  }  while(current!=null){  if((current.data%2)==0){  System.out.print(current.data+" ");  }current=current.next;  }  }  int get(int indeks){  node current = head;  for (int i = 0; i<indeks; i++){  current=current.next;  }  return current.data;  }  void hapustengah(node hapus, node d){  //node hapus = null;  if(head==hapus){  head=hapus.next;  }  if(hapus.next!=null){  hapus.next.prev=hapus.prev;  }  if(hapus.prev!=null){  hapus.prev.next=hapus.next;  }  }  void clear(){  head=tail=null;  System.out.println("data sudah terhapus");  }  void fibonaci() {  int a, b = 1, c=1;    for (int i = 1; i <35; i++) {  a = b;  b = c;  c=a+b;  System.out.print(a+" ");  }  }  public int punyaAkar(int data){  node bantu = head;  while(bantu != null){  int p = 0;  for (int i = 0; i<bantu.data; i++){  if (i\*i==bantu.data){  p++;  }  }  if(p == 1){  System.out.print(bantu.data+ " ");  }  bantu = bantu.next;  } return data;  }  public int prima(int data){  node bantu = head;  while(bantu != null){  int p = 0;  for (int i = 1; i<=bantu.data; i++){  if (bantu.data % i == 0){  p++;  }  }  if(p ==2 ){  System.out.print(bantu.data+ " ");  }  bantu = bantu.next;  } return data;  }  public void print(){  if(isEmpty()){  System.out.println("List masih kosong");  }  else{  node current=head;  do{  System.out.print(current.data+" ");  current=current.next;  }while(current!=null);  }  }  public static void main(String[] args){  tugasmodul2 B = new tugasmodul2();  Random r = new Random();  int angka=0;  for(int i=1; i<66; i++){  angka=r.nextInt(66);  B.addLast(angka);  }  System.out.println("\nAngka Random: ");  B.print();    System.out.println("\nAngka Ganjil: ");  B.ganjil();    System.out.println("\nAngka Genap: ");  B.genap();    System.out.println("\nAngka Fibonaci");  B.fibonaci();    System.out.println("\nAngka Prima: ");  B.prima(angka);    System.out.println("\nAngka yang punya akar: ");  B.punyaAkar(angka);  //System.out.println("\nHapus Semua Data");  //B.clear();  B.print();  }  } |

1. Hasil prorgram

|  |
| --- |
|  |

**TUGAS 1 MODUL III**

*Free Cell*

1. *Source Code*

|  |
| --- |
| import java.lang.Math;  import java.util.Scanner;  //import java.io.IOException;  class Card{  public int data;  public char symbol;  Card next, prev;  public Card(String value){  data = Character.getNumericValue(value.charAt(1));  symbol = value.charAt(0);  next = prev = null;  }  public void show(){  System.out.print("[");  System.out.print(symbol+" "+data+"]");  }  }  class Stack{  char name;  Card head , tail;  Stack next, prev;  public Stack(char name){  this.name = name;  head = tail = null;  next = prev = null;  }  public void add(Card add){  if(head==null) {  head = tail = add;  }  else{  tail.next=add;  add.prev=tail;  tail=add;  }  }  public void push(Card add){  if(head==null) {  head = tail = add;  }  else{  if ((tail.symbol!=add.symbol)&&((tail.data-1)==add.data)){  tail.next=add;  add.prev=tail;  tail=add;  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ INVALID MOVE ]======================================");  }  }  }  public boolean pushHand(Card add){  boolean bool = true;  if(tail==null) {  head = tail = add;  bool = false;  }  else{  if ((head.symbol!=add.symbol)&&((head.data+1)==add.data)){  head.prev=add;  add.next=head;  head=add;  bool = false;  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ INVALID MOVE ]======================================");  }  }  return bool;  }  public Card pop(){  Card current = head;  if(current==null) {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n====================================[ THERE IS NO CARD ]====================================");  return null;  }  else if (head==tail) {  head = tail = null;  }  else{  current = tail;  tail = current.prev;  current.prev = null;  tail.next = null;  }  return current;  }  public void show(){  Card current = head;  System.out.print(name+" : ");  if (current==null){  System.out.println("There is no card!");  } else {  while (current!=null){  current.show();  current = current.next;  }  System.out.println();  }  }  }  public class FreeCell{  Stack hand = new Stack(' ');  Card minus = null;  Card plus = null;  Card[] cell = {null, null, null, null};  Stack head = null, tail = null;  public void add(Stack add){  if(head==null) {  head = tail = add;  }  else{  tail.next = add;  add.prev = tail;  tail = add;  }  }  public Stack getStack(char name){  Stack current = head;  while (current.name!=name){  current = current.next;  }  if (current==null){  System.out.println("\n\n\n\n\n\n\n\n\n\n\n===================================[ THERE IS NO STACK ]===================================");  }  return current;  }  public void show(){  Stack current = head;  if (current==null){  System.out.println("\n\n\n\n\n\n\n\n\n\n\n===================================[ THERE IS NO STACK ]===================================");  } else {  while (current!=null){  current.show();  System.out.println("\n");  current = current.next;  }  }  }  public boolean addMinus(Card add){  boolean bool = true;  if(minus==null){  if(add.symbol=='-'&&add.data==0){  minus = add;  bool = false;  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ INVALID MOVE ]======================================");  }  } else {  if(add.symbol=='-'&&((minus.data+1)==add.data)){  minus = add;  bool = false;  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ INVALID MOVE ]======================================");  }  }  return bool;  }  public boolean addPlus(Card add){  boolean bool = true;  if(plus==null){  if(add.symbol=='+'&&add.data==0){  plus = add;  bool = false;  } else{  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ INVALID MOVE ]======================================");  }  } else {  if(add.symbol=='+'&&((plus.data+1)==add.data)){  plus = add;  bool = false;  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ INVALID MOVE ]======================================");  }  }  return bool;  }  public boolean addCell(Card add){  int index = -1;  boolean bool = true;  for (int i = 0; i<cell.length; i++){  if (cell[i]==null){  index = i;  break;  }  }  if (index==-1){  System.out.println("\n\n\n\n\n\n\n\n\n\n\n======================================[ NO MORE CELL ]=====================================");  } else {  cell[index]=add;  bool = false;  }  return bool;  }  public Card retrieveCell(String card){  Card tmp = null;  int data = Character.getNumericValue(card.charAt(1));  char symbol = card.charAt(0);  int index = -1;  for (int i = 0; i<cell.length; i++){  if (cell[i].data==data&&cell[i].symbol==symbol){  index = i;  break;  }  }  if (index==-1){  System.out.println("Cannot find the card!");  } else {  tmp = cell[index];  cell[index]= null;  }  return tmp;  }  public void showGame(){  System.out.println("\n\n\n\n\n\n\n\n========================================[ FREECELL ]========================================");  System.out.println();  for(int i = 0; i<4; i++){  System.out.print("--------------");  if (cell[i]==null){  System.out.print("[ ]");  } else {  cell[i].show();  }  }  System.out.println("--------------");  System.out.println();  System.out.print("Plus Cards : ");  if (plus==null){  System.out.print("[ ]");  } else {  plus.show();  }  System.out.print("\nMinus Cards : ");  if (minus==null){  System.out.print("[ ]");  } else {  minus.show();  }  System.out.println("\n------------------------------------------------------------------------------------------");  show();  System.out.println("\n------------------------------------------------------------------------------------------");  System.out.print("\nOn your hand : ");  hand.show();  System.out.println("\n========================================[ FREECELL ]========================================");  checkPoint();  }  public void checkPoint(){  if(plus!=null&&minus!=null){  if (plus.data==9&&minus.data==9){  for(int i = 0; i<150; i++){  System.out.println("\r\n");  }  System.out.println("\n========================================[ FREECELL ]========================================");  System.out.println("\n============================================================================================");  System.out.println("\n=====================================[ CONGRATULATION ]=====================================");  System.out.println("\n============================================================================================");  System.out.println("\n====================================[ YOU WON THE GAME ]====================================");  System.out.println("\n============================================================================================");  }  }  }  public static void main(String[] args) {  Scanner cin = new Scanner(System.in);  FreeCell game = new FreeCell();  Stack stack\_tmp = null;  Card card\_tmp = null, add;  game.add(new Stack('A'));  game.add(new Stack('B'));  game.add(new Stack('C'));  game.add(new Stack('D'));  game.add(new Stack('E'));  game.add(new Stack('F'));  game.add(new Stack('G'));  String[] deck = {"+7","+8","+4","+0","+1","+5","-4","+2","-3","-0","+9","+3","-7","+6","-8","-5","-1","-2","-9","-6"};  char name\_stack = 'A';  String data\_card = "10";  boolean save = true;  stack\_tmp = game.getStack(name\_stack);  int i = deck.length-1, choice=0;  while(i>=0){  if ((i)%3==0&&i!=0){  name\_stack++;  //System.out.print(name\_stack);  stack\_tmp = game.getStack(name\_stack);  }  //System.out.print(deck[i]);  stack\_tmp.add(new Card(deck[i]));  i--;  }  do{  game.showGame();  System.out.println("\nMake a move!");  System.out.println("1. Get a card.\n2. Put cards");  System.out.println("3. Collect a card.\n4. Throw a card");  System.out.println("5. Retrieve a card\n0. Quit");  System.out.print("Your choice : ");  choice = cin.nextInt();  switch (choice){  case 1 :  System.out.print("Line name : ");  name\_stack = cin.next().charAt(0);  if(name\_stack>90){  name\_stack-= 32;  }  stack\_tmp = game.getStack(name\_stack);  if (stack\_tmp!=null){  card\_tmp = stack\_tmp.pop();  if (card\_tmp!=null){  if (game.hand.pushHand(card\_tmp)){  stack\_tmp.add(card\_tmp);  }  }  } break;  case 2 :  System.out.print("Line name : ");  name\_stack = cin.next().charAt(0);  if(name\_stack>90){  name\_stack-= 32;  }  stack\_tmp = game.getStack(name\_stack);  if (stack\_tmp!=null){  if (game.hand.head!=null){  if(stack\_tmp.head==null){  stack\_tmp.head = game.hand.head;  stack\_tmp.tail = game.hand.tail;  } else {  stack\_tmp.tail.next = game.hand.head;  game.hand.head.prev = stack\_tmp.tail;  stack\_tmp.tail = game.hand.tail;  }  game.hand.head = game.hand.tail = null;  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n====================================[ GET A CARD FIRST! ]===================================");  }  }break;  case 3 :  System.out.print("Line name : "'');  name\_stack = cin.next().charAt(0);  if(name\_stack>90){  name\_stack-= 32;  }  stack\_tmp = game.getStack(name\_stack);  if (stack\_tmp!=null){  card\_tmp = stack\_tmp.pop();  if (card\_tmp!=null){  if(card\_tmp.symbol=='+'){  if(game.addPlus(card\_tmp)){  stack\_tmp.add(card\_tmp);  }  } else if (card\_tmp.symbol=='-'){  if(game.addMinus(card\_tmp)){  stack\_tmp.add(card\_tmp);  }  } else {  stack\_tmp.add(card\_tmp);  }  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n====================================[ GET A CARD FIRST! ]===================================");  }  } break;  case 4 :  System.out.print("Line name : ");  name\_stack = cin.next().charAt(0);  if(name\_stack>90){  name\_stack-= 32;  }  stack\_tmp = game.getStack(name\_stack);  if (stack\_tmp!=null){  card\_tmp = stack\_tmp.pop();  if (card\_tmp!=null){  if(game.addCell(card\_tmp)){  stack\_tmp.push(card\_tmp);  }  } else {  System.out.println("\n\n\n\n\n\n\n\n\n\n\n====================================[ GET A CARD FIRST! ]===================================");  }  } break;  case 5 :  System.out.print("Choose card : ");  data\_card = cin.nextLine();  data\_card = cin.nextLine();  card\_tmp = game.retrieveCell(data\_card);  if (card\_tmp!=null){  if(game.hand.pushHand(card\_tmp)){  save = game.addCell(card\_tmp);  }  } break;  case 0 :  for(int c = 0; c<150; c++){  System.out.println("\r\n");  }  System.out.println("\n========================================[ FREECELL ]========================================");  System.out.println("\n============================================================================================");  System.out.println("\n=======================================[ GAME OVER ]=======================================");  System.out.println("\n============================================================================================");  System.out.println("\n========================================[ YOU LOSE ]========================================");  System.out.println("\n============================================================================================");  return;  default : System.out.println("option not available!");  }  }while(choice!=0);}} |

**TIKET RESPON MODUL III**

Kalkulator dengan *Stack*

1. *Source Code*

|  |
| --- |
| import java.util.Scanner;  class Node {  public Node next, prev;  char data;  public Node(char data) {  this.data = data;}}  class Stack {  Node head, tail;  int size = 0;    public void push(char data) {  Node newest = new Node(data);  if(head==null) {  head = tail = newest;}  else {  head.prev = newest;  newest.next = head;  head = newest;}  size++;}  public char pop() {  char top=peek();  if(head==null){  System.out.println("Linked List masih kosong!");}  else{  if(tail==head){  tail=head=null;}  else{  Node current = head;  head = current.next;  current = null;}}  size--;  return top;}    public char peek() {  return head.data;}}  public class kalkulator {  int hasilExp = 0;  Stack operand = new Stack();  Stack operator = new Stack();  char a,b;  int c;  public boolean isOperator(char ch){  if (ch=='+' || ch=='-' || ch=='\*' || ch=='/' || ch=='(')  return true;  return false;}    public void calculate(char ch){  if(ch=='+'){  b=operand.pop(); a=operand.pop(); c=((int)a-48)+((int)b-48);}  else if(ch=='-'){  b=operand.pop(); a=operand.pop(); c=((int)a-48)-((int)b-48);}  else if(ch=='\*'){  b=operand.pop(); a=operand.pop(); c=((int)a-48)\*((int)b-48);}  else{  b=operand.pop(); a=operand.pop(); c=((int)a-48)/((int)b-48);}  operand.push((char)(c+48));}    public int execute(String infixExp){  for (int i = 0; i < infixExp.length(); i++){  char ch = infixExp.charAt(i);    if(isOperator(ch))  operator.push(ch);  else if(ch==')'){  do{  calculate(operator.pop());}  while (operator.peek()!= '(');  operator.pop();}  else  operand.push(ch);  if(operand.size==2 && operator.size==1)  calculate(operator.pop());}  return ((int)operand.pop()-48);}    public static void main(String[] args){  kalkulator noer = new kalkulator();  Scanner in = new Scanner(System.in);  System.out.print("Infix Expression : ");  String infix = in.nextLine();  System.out.println("Hasil : " +noer.execute(infix));}} |

1. Hasil program

|  |
| --- |
|  |

**TUGAS 1 MODUL IV**

Program *Binary Search*

* + - * 1. Algoritma
      1. Mengurutkan *array* data yang telah dimasukkan secara statis
      2. Membuat *method* pencarian untuk data yang sudah terurut
      3. Untuk melakukan proses *binary searching*, diperlukan untuk membagi dua data agar mendapat nilai tengah sebagai pembanding
      4. Melakukan perulangan selama indeks pengecekan kurang dari sama dengan indeks akhir *array*.
      5. Jika data yang dicari sama dengan nilai tengah, maka akan dikeluarkan nilai tengah
      6. Jika data yang dicari lebih kecil dari nilai tengah, maka pengecekan akan berjalan ke kiri nilai tengah
      7. Selain itu, maka pencarian akan berjalan ke kanan nilai tengah

1. *Source Code*

|  |
| --- |
| public class binarysearch {    static void bubblesort(int[]angka){  for(int i=0; i<angka.length; i++){  for(int j = 0; j<angka.length-1; j++){  if(angka[j]>angka[j+1]){  int temp = angka[j];  angka[j]=angka[j+1];  angka[j+1]=temp;}}}  }  static int binary(int[]angka, int cari, int i, int j){  int tengah;  while (i<=j){  tengah = (i+j)/2;  if(cari==angka[tengah]){  return tengah;  }  else if(cari<angka[tengah]){  j=tengah-1;  }else{  i=tengah+1;  }  }return -1;  }    public static void main(String[] args){  int[]angka = {1,75,68,9,55,20,99};  bubblesort(angka);  System.out.println("Setelah diurutkan:");  for(int i=0; i<angka.length; i++){  System.out.print(angka[i]+" ");}    int cari = 99;    int hasil = binary(angka,cari,0,angka.length-1);  System.out.println("");  if(hasil>=0){  System.out.println("Angka ketemu di indeks ke-"+hasil);  }else{  System.out.println("Angka tidak ada.");}}} |

Program *Sorting*

1. *Quick sort*

Algoritma

1. Menetapkan nilai bawah dna nilai atas sebuah *array*
2. Menetapkan pivot di nilai bawah *array*
3. Melakukan perulangan sekali, sebelum memenuhi syarat selama nilai bawah kurang dari sama dengan nilai atas
4. Selama nilai pada indeks bawah kurang dari pivot, maka pengecekan pada indeks bawah akan pindah ke sebelah kanannya
5. Selama nilai pada indeks atas lebih besar dari pivot, maka pengecekan pada indeks atas akan pindah ke sebelah kirinya
6. Jika nilai indeks bawah kurang dari sama dengan nilai indeks atas
7. Memindahkan nilai indeks bawah, lalu menukarnya dengan nilai indeks atas
   1. *Source Code*

|  |
| --- |
| public class quicksort{  static void quickSort (int a[], int lo, int hi){  int i=lo, j=hi, h;  int pivot=a[lo];  do{  while (a[i]<pivot) i++;  while (a[j]>pivot) j--;  if (i<=j){  h=a[i]; a[i]=a[j];  a[j]=h;//tukar  i++;  j--;}}  while (i<=j);  // pengurutan  if (lo<j) quickSort(a, lo, j);  if (i<hi) quickSort(a, i, hi);}  public static void main(String[] args) {  int angka[]={1,75,68,9,55,20,99};  int i,n=7;  System.out.print("data sebelum di urutkan\n");  for(i=0;i<n;i++){  System.out.print(angka[i]+ " ");  }  System.out.print("\n");  quickSort(angka,0,n-1);  System.out.print("\nsetelah di urutkan dengan quick sort\n");  for(i=0;i<n;i++){  System.out.print(angka[i]+" ");}}} |

1. *Merge sort*

Algoritma

1. Menetapkan nilai bawah dan nilai atas sebuah *array*.
2. Jika nilai bawah lebih besar nilai atas maka akan melakukan proses *merge sort*.
3. Membagi *array* menjadi dua bagian.
4. Mengurutkan kedua bagian dengan cara rekursif.
5. Membandingkan besar nilai yang di kiri dan kanan
6. Jika nilai di kiri lebih keci dari kanan, maka posisi tetap dan dilakukan perbandingan di data selanjutnya
7. Jika nilai di kiri lebih besar dari kanan, maka posisi akan ditukar dan perbandingan dipindah ke data sebelahnya
8. Menggabung keduanya kembali

*Source Code*

|  |
| --- |
| public class mergesort {  public static void mergeS(int n[], int l, int h){  int low = l;  int high = h;  if (low>=high){  return;  }  int middle=(low+high)/2;  mergeS(n,low,middle);  mergeS(n,middle+1,high);  int end\_low=middle;  int start\_high=middle+1;  while ((l<=end\_low) && (start\_high<=high)){  if (n[low]<n[start\_high]){  low++;  }  else{  int Temp=n[start\_high];  for (int k=start\_high-1;k>=low;k--){  n[k+1]=n[k];  }  n[low]=Temp;  low++;  end\_low++;  start\_high++;  }  }  }  public static void main(String[]args){    int n[] ={2371,2,8,383,67,39,80,362,0};  System.out.println("Sebelum di Urutkan = ");  for(int i = 0; i < n.length; i++){  System.out.print(n[i]+" ");  }  System.out.println();    mergeS(n,0,n.length-1);  System.out.println();  System.out.println("Setelah di Urutkan = ");  for(int i = 0; i < n.length; i++){  System.out.print(n[i]+" "); } }} |

**TUGAS 1 MODUL VI**

Program BFS dan DFS

1. *Source Code*

|  |
| --- |
| import java.util.Scanner;  class node{  int data;  boolean ada;  edge ke;  node next;  node(int a){  data=a;  }  node(int a, edge b){  data = a; ke = b;  }  node(node a){  data = a.data;  ke = a.ke;  }  }  class edge{  node tujuan;  edge lanjut;  edge(node b){  tujuan = b;  }  }  class list{  node head, tail;  void tambah(int a){  node baru = new node(a);  if(head==null){  head = tail = baru;  }  else {  tail.next = baru;  tail = baru;  }  }    void hubung(int a, int b){  node bantu = head;  while(bantu.data!=a){  bantu = bantu.next;  }    hubungedge(bantu,b);    }    void hubungedge(node bantu, int b){  node bantu2 = head;  while(bantu2.data!=b){  bantu2=bantu2.next;  }  if(bantu.ke ==null){bantu.ke=new edge(bantu2);}  else{  edge baru = bantu.ke;  while(baru.lanjut!=null){  baru = baru.lanjut;  }  baru.lanjut = new edge(bantu2);  }  }    void adjlist(){  node bantu = head;  while(bantu!=null){  System.out.print(bantu.data);  edge lihat = bantu.ke;  while(lihat!=null){  System.out.print(" -> "+lihat.tujuan.data);  lihat = lihat.lanjut;  }  System.out.println("");  bantu = bantu.next;  }  }  }  class stack{  node top;  void tambah(node a){  node baru = new node(a);  if (top==null){  top = baru;  }  else {  baru.next = top;  top = baru;  }  }    node pop(){  if(top==null){return null;}  node tmp = top;  top = top.next;  return tmp;  }  }  class queue{  node front, tail;  void tambah(node a){  node baru = new node(a.data, a.ke);  if (front==null){  front = tail = baru;  }  else {  tail.next = baru;  tail = baru;  }  }    node dequeue(){  if(front==null){return null;}  node tmp = front;  front = front.next;  return tmp;  }  }  public class responmod6{  public static void main(String[] args){  Scanner input = new Scanner(System.in);  list daftar = new list();  int i, a;  for(i=0;i<=6;i++){  daftar.tambah(i);  }  daftar.hubung(0, 1);  daftar.hubung(1, 2);  daftar.hubung(1, 5);  daftar.hubung(2, 3);  daftar.hubung(2, 0);  daftar.hubung(3, 4);  daftar.hubung(4, 6);  daftar.hubung(4, 0);  daftar.hubung(5, 6);  System.out.println("adjacency list:");  daftar.adjlist();  System.out.print("vertex sumber bfs: ");  a = input.nextInt();    bfs(a, daftar);    System.out.print("\nvertex sumber dfs: ");  a = input.nextInt();  dfs(a, daftar);  System.out.println("");    }    static node cari(int a, list daftar){  node bantu3 = daftar.head;  while (bantu3!=null){  if(bantu3.data==a){return bantu3;}  bantu3 = bantu3.next;  }  return bantu3;  }  static void dfs(int a, list daftar){  stack stek = new stack();  node bantu = cari(a, daftar);  int[] sudah = new int[7];  int x, i;  System.out.print("Pencarian dfs: ");  while(bantu!=null){  x=0;  if(sudah[bantu.data]==0){  sudah[bantu.data] = 1;  System.out.print(bantu.data);  for(i=0;i<=6;i++){  x += sudah[i];}  if(x==7){return;}  edge lihat = bantu.ke;  while(lihat!=null){  stek.tambah(lihat.tujuan);  lihat = lihat.lanjut;  }  }  else{}  bantu = stek.pop();  if(bantu!=null&&sudah[bantu.data]!=1){  System.out.print(" -> ");}  }  }    static void bfs(int a, list daftar){  queue que = new queue();  node bantu2 = cari(a, daftar);  int[] sudah = new int[7];  int x, i;  System.out.print("Pencarian bfs: ");  while(bantu2!=null){  x=0;  if(sudah[bantu2.data]==0){  sudah[bantu2.data] = 1;  System.out.print(bantu2.data);  for(i=0;i<=6;i++){  x += sudah[i];  }  if(x==7){return;}  edge lihat = bantu2.ke;  while(lihat!=null){  que.tambah(lihat.tujuan);  lihat = lihat.lanjut;  }  }  else{}  if (que.front==null){return;}  bantu2 = que.dequeue();  if(bantu2!=null&&sudah[bantu2.data]!=1){System.out.print(" -> ");}  }  }  } |

1. Hasil program

|  |
| --- |
|  |

**TUGAS 1 MODUL VII**

Program Instagram

1. *Source Code*

|  |
| --- |
| import java.util.Scanner;  class node{  int data;  String nama;  boolean ada;  edge ke;  node next;  node(int a){data=a;}  // node(){}  node(int a, edge b){  data = a; ke = b;  }  node(node a){  data = a.data;  ke = a.ke;  }  node (int a, String b){data=a; nama=b; ada=false;}  }  class edge{  node tujuan;  edge lanjut;  edge(node b){  tujuan = b;  }  }  class list{  node head, tail;  void tambah(int a, String b){  node baru = new node(a, b);  if(head==null){  head = tail = baru;  }  else {  tail.next = baru;  tail = baru;  }  }    void hubung(int a, int b){  node bantu = head;  while(bantu.data!=a){  bantu = bantu.next;  }    hubungedge(bantu,b);    }  node search(int a){  node bantu=head;  while(bantu!=null){  if(bantu.data==a){return bantu;}  bantu=bantu.next;  }  System.out.println("Tidak ditemukan");  return bantu;  }    node search(String a){  node bantu=head;  while(bantu!=null){  if(a.equals(bantu.nama)){return bantu;}  bantu=bantu.next;  }  System.out.println("Tidak ditemukan");  return null;  }    void hubungedge(node bantu, int b){  node bantu2 = head;  while(bantu2.data!=b){  bantu2=bantu2.next;  }  if(bantu.ke ==null){bantu.ke=new edge(bantu2);}  else{  edge baru = bantu.ke;  while(baru.lanjut!=null){  baru = baru.lanjut;  }  baru.lanjut = new edge(bantu2);  }  }    void tampil(){  node bantu = head;  while (bantu!=null){  System.out.println(bantu.data);  bantu = bantu.next;  }  }    void adjlist2(){  node bantu = head;  while(bantu!=null){  System.out.print(bantu.data+"/"+bantu.nama);  edge lihat = bantu.ke;  while(lihat!=null){  System.out.print(" -> "+lihat.tujuan.data+"/"+lihat.tujuan.nama);  lihat = lihat.lanjut;  }  System.out.println("");  bantu = bantu.next;  }  }  }  public class tugasmod7 {  public static void main(String[] args){  Scanner input = new Scanner (System.in);  String nama;  int a;  list daftar = new list();  daftar.tambah(0, "Kiya");  daftar.tambah(1, "Bella");  daftar.tambah(2, "Fira");  daftar.tambah(3, "Angga");  daftar.tambah(4, "Bim");  daftar.tambah(5, "Naim");  daftar.tambah(6, "Farid");  daftar.tambah(7, "Fariz");  daftar.tambah(8, "Richo");  daftar.tambah(9, "Meong");  daftar.hubung(0, 1);  daftar.hubung(1, 2);  daftar.hubung(1, 5);  daftar.hubung(2, 3);  daftar.hubung(2, 0);  daftar.hubung(3, 4);  daftar.hubung(4, 6);  daftar.hubung(4, 0);  daftar.hubung(4, 5);  daftar.hubung(5, 4);  daftar.hubung(5, 6);  daftar.hubung(5, 7);  daftar.hubung(6, 7);  daftar.hubung(7, 8);  daftar.hubung(8, 9);  daftar.hubung(9, 6);  daftar.adjlist2();  while  System.out.println("Login:");  System.out.print("Masukkan id: ");  a=input.nextInt();  node x = daftar.search(a);  System.out.println("Login sebagai "+ x.nama);  edge follow = x.ke;  System.out.println("Following:");  while(follow!=null){  System.out.println(" - "+follow.tujuan.nama);  follow = follow.lanjut;  }    System.out.println("Follower: ");  follower(daftar,x);  // System.out.println("mutual");  // tf(daftar,x);  int id;  int sesama=0;  String namaa;  input.nextLine();  System.out.print("Nama yang ingin dicari: ");  namaa = input.nextLine();  // System.out.println(id);  node y = daftar.search(namaa);  if(y==null){System.out.println("Tidak ada"); return;}  node bantu = y;  edge temanx=x.ke;  boolean sumber=false;  boolean tujuan=false;  boolean mutual=false;  while(temanx!=null){  if(temanx.tujuan==y){sumber=true;}  temanx=temanx.lanjut;  }    edge temany=y.ke;  while(temany!=null){  if(temany.tujuan==x){tujuan=true;}  temany=temany.lanjut;  }  if(sumber&&tujuan){System.out.println("Saling follow: "+y.nama);}  temanx=x.ke;  while(temanx!=null){  temany=y.ke;  while(temany!=null){  node temanz = temany.tujuan;  if(temany.tujuan==temanx.tujuan){mutual=true;}  if(mutual){System.out.println("Mutual: "+temanz.nama);mutual=false;}    temany=temany.lanjut;  }  temanx=temanx.lanjut;  }    System.out.println("Explore:");  explore(daftar,x);    System.out.println("Tambah teman: ");  System.out.print("Nama yang ingin dicari: ");  namaa = input.nextLine();  y=daftar.search(namaa);  daftar.hubung(x.data, y.data);  daftar.adjlist2();  }    static void follower(list daftar, node x){  node bantu=daftar.head;  while(bantu!=null){  if(bantu==x){bantu=bantu.next;  if(bantu==null){break;}  }  edge bantu2 = bantu.ke;  while(bantu2!=null){  if(bantu2.tujuan==x){System.out.println(" - "+bantu.nama);}  bantu2=bantu2.lanjut;  }  bantu=bantu.next;  }  }    static void tf(list daftar, node x){  Scanner masuk = new Scanner(System.in);  int id;  System.out.print("Nama yang ingin dicari: ");  id = masuk.nextInt();  System.out.println(id);  node y = daftar.search(id);  if(y==null){System.out.println("Tidak ada"); return;}  node bantu = daftar.head;  System.out.println(y.nama);  while(bantu!=null){  if(bantu==x||bantu==y){bantu=bantu.next;  if(bantu==null){break;}  }  edge bantu2 = bantu.ke;  while(bantu2!=null){  if(bantu2.tujuan==y){System.out.println("Followed by: "+ bantu.nama);}  bantu2=bantu2.lanjut;  }  }  }    static void explore(list daftar, node x){  node bantu = x;  edge bantu2 = bantu.ke;  while(bantu2!=null){  bantu2.tujuan.ada = true;  bantu2 = bantu2.lanjut;  }    bantu = daftar.head;  while(bantu!=null){  if(bantu==x){bantu=bantu.next;  if(bantu==null){break;}  }  if(!bantu.ada){  System.out.println(" - "+ bantu.nama);  bantu.ada=false;  }  bantu=bantu.next;  }  }  } |

1. Hasil program

|  |
| --- |
|  |