|  |
| --- |
| 1. **Design suitable data structures and implement pass-I of a two-pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.** |

import java.io.\*;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

class searchAd

{

int arr\_length;

boolean searchad(String []s\_arr,int lc1) throws IOException

{

arr\_length = s\_arr.length;

try(FileWriter fw = new FileWriter("assemblertbl/symtbl.txt", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter out = new PrintWriter(bw))

{

if(s\_arr[1].equalsIgnoreCase("DC")||s\_arr[1].equalsIgnoreCase("DS"))

{

out.println(s\_arr[0]+" "+lc1+" 1");

}

}

catch(Exception e)

{

}

try(FileWriter fw1 = new FileWriter("assemblertbl/optblout.txt", true);

BufferedWriter bw1 = new BufferedWriter(fw1);

PrintWriter outop = new PrintWriter(bw1))

{

try(FileWriter fw2 = new FileWriter("assemblertbl/symtbl.txt", true);

BufferedWriter bw2 = new BufferedWriter(fw2);

PrintWriter outsym = new PrintWriter(bw2))

{

if(arr\_length==4)

{

outsym.println(s\_arr[0]+" "+lc1+" 1");

BufferedReader br=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblin.txt")));

String s;

while((s=br.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr1[]=new String[st.countTokens()];

for(int i=0;i<s\_arr1.length;i++)

{

s\_arr1[i]=st.nextToken();

}

for(int j=0;j<s\_arr1.length;j++)

{

if(s\_arr[1].equalsIgnoreCase(s\_arr1[j]))

{

outop.println(s\_arr1[j]+" "+s\_arr1[j+1]+" 1");

}

}

}

}

if(arr\_length==3||arr\_length==2||arr\_length==1)

{

BufferedReader br=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblin.txt")));

String s;

while((s=br.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr1[]=new String[st.countTokens()];

for(int i=0;i<s\_arr1.length;i++)

{

s\_arr1[i]=st.nextToken();

}

for(int j=0;j<s\_arr1.length;j++)

{

if(s\_arr[0].equalsIgnoreCase(s\_arr1[j]))

{

outop.println(s\_arr1[j]+" "+s\_arr1[j+1]+" 1");

}

}

}

}

}

catch (IOException e)

{

}

}

catch (IOException e)

{

}

return true;

}

void varient(String s\_arr[],int loc) throws FileNotFoundException, IOException

{

int arr\_length1=s\_arr.length;

try(FileWriter fw = new FileWriter("assemblertbl/output.txt", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter out = new PrintWriter(bw))

{

if(s\_arr[0].equalsIgnoreCase("START"))

{

out.println("AD 01 C "+(loc+1));

}

if(s\_arr[0].equalsIgnoreCase("END"))

{

out.println("AD 02");

}

if(arr\_length1==1)

{

BufferedReader bf1=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblout.txt")));

String s;

while((s=bf1.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr1[]=new String[st.countTokens()];

for(int i=0;i<s\_arr1.length;i++)

{

s\_arr1[i]=st.nextToken();

}

if(s\_arr[0].equalsIgnoreCase(s\_arr1[0]))

{

out.println("IS "+s\_arr1[1]);

}

}

}

if(arr\_length1==2)

{

BufferedReader bf1=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblout.txt")));

String s;

while((s=bf1.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr1[]=new String[st.countTokens()];

for(int i=0;i<s\_arr1.length;i++)

{

s\_arr1[i]=st.nextToken();

}

if(s\_arr[0].equalsIgnoreCase(s\_arr1[0]))

{

if(s\_arr[1].equalsIgnoreCase("AREG"))

{

out.println("IS "+s\_arr1[1]+" 1");

}

if(s\_arr[1].equalsIgnoreCase("BREG"))

{

out.println("IS "+s\_arr1[1]+" 2");

}

if(s\_arr[1].equalsIgnoreCase("CREG"))

{

out.println("IS "+s\_arr1[1]+" 3");

}

if(s\_arr[1].equalsIgnoreCase("DREG"))

{

out.println("IS "+s\_arr1[1]+" 4");

}

BufferedReader bf2=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/symtbl.txt")));

String s2;

while((s2=bf2.readLine())!=null)

{

StringTokenizer st2=new StringTokenizer(s2," ",false);

String s\_arr2[]=new String[st2.countTokens()];

for(int i=0;i<s\_arr2.length;i++)

{

s\_arr2[i]=st2.nextToken();

}

if(s\_arr[1].equalsIgnoreCase(s\_arr2[0]))

{

out.println("IS "+s\_arr1[1]+" "+s\_arr2[1]);

}

}

}

}

}

if(arr\_length1==3)

{

if(s\_arr[1].equalsIgnoreCase("DS"))

{

out.println("DL 02 C "+s\_arr[2]);

}

if(s\_arr[1].equalsIgnoreCase("DC"))

{

out.println("DL 01 C "+s\_arr[2]);

}

}

if(arr\_length1==3)

{

BufferedReader bf1=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblout.txt")));

String s;

while((s=bf1.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr1[]=new String[st.countTokens()];

for(int i=0;i<s\_arr1.length;i++)

{

s\_arr1[i]=st.nextToken();

}

if(s\_arr[0].equalsIgnoreCase(s\_arr1[0]))

{

BufferedReader bf2=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/symtbl.txt")));

String s2;

while((s2=bf2.readLine())!=null)

{

StringTokenizer st2=new StringTokenizer(s2," ",false);

String s\_arr2[]=new String[st2.countTokens()];

for(int i=0;i<s\_arr2.length;i++)

{

s\_arr2[i]=st2.nextToken();

}

if(s\_arr[2].equalsIgnoreCase(s\_arr2[0]))

{

if(s\_arr[1].equalsIgnoreCase("AREG"))

{

out.println("IS "+s\_arr1[1]+" 1"+" S "+s\_arr2[1]);

}

if(s\_arr[1].equalsIgnoreCase("BREG"))

{

out.println("IS "+s\_arr1[1]+" 2"+" S "+s\_arr2[1]);

}

if(s\_arr[1].equalsIgnoreCase("CREG"))

{

out.println("IS "+s\_arr1[1]+" 3"+" S "+s\_arr2[1]);

}

if(s\_arr[1].equalsIgnoreCase("DREG"))

{

out.println("IS "+s\_arr1[1]+" 4"+" S "+s\_arr2[1]);

}

}

}

}

}

}

if(arr\_length1==4)

{

BufferedReader bf1=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblout.txt")));

String s;

while((s=bf1.readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr1[]=new String[st.countTokens()];

for(int i=0;i<s\_arr1.length;i++)

{

s\_arr1[i]=st.nextToken();

}

if(s\_arr[1].equalsIgnoreCase(s\_arr1[0]))

{

BufferedReader bf2=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/symtbl.txt")));

String s2;

while((s2=bf2.readLine())!=null)

{

StringTokenizer str2=new StringTokenizer(s2," ",false);

String s\_arr2[]=new String[str2.countTokens()];

for(int j=0;j<s\_arr2.length;j++)

{

s\_arr2[j]=str2.nextToken();

}

if(s\_arr[3].equalsIgnoreCase(s\_arr2[0]))

{

if(s\_arr[2].equalsIgnoreCase("AREG"))

{

out.println("IS "+s\_arr1[1]+" 1"+" S "+s\_arr2[1]);

}

if(s\_arr[2].equalsIgnoreCase("BREG"))

{

out.println("IS "+s\_arr1[1]+" 2"+" S "+s\_arr2[1]);

}

if(s\_arr[2].equalsIgnoreCase("CREG"))

{

out.println("IS "+s\_arr1[1]+" 3"+" S "+s\_arr2[1]);

}

if(s\_arr[2].equalsIgnoreCase("DREG"))

{

out.println("IS "+s\_arr1[1]+" 4"+" S "+s\_arr2[1]);

}

}

}

}

}

}

}

catch (IOException e)

{

}

}

}

class passoneofassembler

{

public static void main(String a[]) throws IOException

{

searchAd search=new searchAd();

int lc1 = 0;

PrintWriter outsym =new PrintWriter(new FileWriter ("assemblertbl/symtbl.txt"),true);

PrintWriter out2 =new PrintWriter(new FileWriter ("assemblertbl/output2.txt"),true);

PrintWriter out =new PrintWriter(new FileWriter ("assemblertbl/output.txt"),true);

PrintWriter optbl =new PrintWriter(new FileWriter ("assemblertbl/optblout.txt"),true);

BufferedReader input=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/input.txt")));

BufferedReader input2=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblin.txt")));

String s;

try(FileWriter fw12 = new FileWriter("assemblertbl/optblout.txt", true);

BufferedWriter bw12 = new BufferedWriter(fw12);

PrintWriter outop2 = new PrintWriter(bw12))

{

outop2.println("opname opcode Length");

}catch(Exception e)

{

}

try(FileWriter fw12 = new FileWriter("assemblertbl/symtbl.txt", true);

BufferedWriter bw12 = new BufferedWriter(fw12);

PrintWriter outop2 = new PrintWriter(bw12))

{

outop2.println("SymName Address Length");

}catch(Exception e)

{

}

while((s=input .readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr[]=new String[st.countTokens()];

for(int i=0;i<s\_arr.length;i++)

{

s\_arr[i]=st.nextToken();

}

if(s\_arr[0].equalsIgnoreCase("START"))

{

lc1=Integer.parseInt(s\_arr[1]);

--lc1;

}

search.searchad(s\_arr,lc1);

lc1++;

}

BufferedReader input3=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/input.txt")));

lc1=0;

while((s=input3 .readLine())!=null)

{

StringTokenizer st=new StringTokenizer(s," ",false);

String s\_arr[]=new String[st.countTokens()];

for(int i=0;i<s\_arr.length;i++)

{

s\_arr[i]=st.nextToken();

}

if(s\_arr[0].equalsIgnoreCase("START"))

{

lc1=Integer.parseInt(s\_arr[1]);

--lc1;

}

search.varient(s\_arr,lc1);

lc1++;

}

BufferedReader bf1=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/input.txt")));

String s1;

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pass 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.out.println("\nInput Table(Assembler Program):\n");

while((s1=bf1.readLine())!=null)

{

System.out.println(s1);

}

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

BufferedReader bf2=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/optblout.txt")));

String s2;

System.out.println("\nOpcode Table:\n");

while((s2=bf2.readLine())!=null)

{

System.out.println(s2);

}

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

BufferedReader bf3=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/symtbl.txt")));

String s3;

System.out.println("\nSymbol Table:\n");

while((s3=bf3.readLine())!=null)

{

System.out.println(s3);

}

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

BufferedReader bf4=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/output.txt")));

String s4;

System.out.println("\nIntermediate Code Generated:\n");

while((s4=bf4.readLine())!=null)

{

System.out.println(s4);

}

}

}

**OUTPUT**

|  |
| --- |
| 1. **Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.** |

import java.io.\*;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

class pass2

{

void target()

{

PrintWriter out = null;

try {

int arr\_length;

out = new PrintWriter(new FileWriter("assemblertbl/output2.txt"));

BufferedReader bf= new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/output.txt")));

String s;

while((s=bf.readLine())!=null)

{

StringTokenizer str=new StringTokenizer(s," ",false);

String s\_arr[]=new String[str.countTokens()];

for(int i=0;i< s\_arr.length;i++)

{

s\_arr[i]=str.nextToken();

}

arr\_length=s\_arr.length;

if(s\_arr[0].equalsIgnoreCase("AD") ||s\_arr[0].equalsIgnoreCase("DL"))

{

continue;

}

else if(arr\_length==5)

{

out.println("+ "+s\_arr[1] +" "+s\_arr[2]+ " "+s\_arr[4]);

}

else if(arr\_length==3)

{

out.println("+ "+s\_arr[1]+" "+s\_arr[2]+" 000");

}

else if(arr\_length==2)

{

out.println("+ "+s\_arr[1]+" 0"+" 000");

}

}

} catch (IOException ex) {

Logger.getLogger(pass2.class.getName()).log(Level.SEVERE, null, ex);

} finally {

out.close();

}

}

}

class passtwooftheassembler

{

public static void main(String a[]) throws IOException

{

pass2 p2 =new pass2();

p2.target();

BufferedReader bf5=new BufferedReader(new InputStreamReader(new FileInputStream("assemblertbl/output2.txt")));

String s5;

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pass 2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.out.println("\nTarget Programm:\n");

while((s5=bf5.readLine())!=null)

{

System.out.println(s5);

}

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

}

}

**OUTPUT**

|  |
| --- |
| 1. **Design suitable data structures and implement pass-I of a two-pass macro-processor using OOP features in Java**   import java.util.\*;  import java.io.\*;  class MntTuple {  String name;  int index;    MntTuple(String s, int i) {  name = s;  index = i;  }    public String toString() {  return("[" + name + ", " + index + "]");  }  }  class MacroProcessor {  static List<MntTuple> mnt;  static List<String> mdt;  static int mntc;  static int mdtc;  static int mdtp;  static BufferedReader input;  static List<List <String>> ala;  static Map<String, Integer> ala\_macro\_binding;    public static void main(String args[]) throws Exception {  initializeTables();  System.out.println("===== PASS 1 =====\n");  pass1();    }    static void pass1() throws Exception {  String s = new String();  input = new BufferedReader(new InputStreamReader(new FileInputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/input.txt")));  PrintWriter output = new PrintWriter(new FileOutputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/output\_pass1.txt"), true);  while((s = input.readLine()) != null) {  if(s.equalsIgnoreCase("MACRO")) {  processMacroDefinition();  } else {  output.println(s);  }  }  System.out.println("ALA:");  showAla(1);  System.out.println("\nMNT:");  showMnt();  System.out.println("\nMDT:");  showMdt();  }    static void processMacroDefinition() throws Exception {  String s = input.readLine();  String macro\_name = s.substring(0, s.indexOf(" "));  mnt.add(new MntTuple(macro\_name, mdtc));  mntc++;  pass1Ala(s);  StringTokenizer st = new StringTokenizer(s, " ,", false);  String x = st.nextToken();  for(int i=x.length() ; i<12 ; i++) {  x += " ";  }  String token = new String();  int index;  token = st.nextToken();  x += token;  while(st.hasMoreTokens()) {  token = st.nextToken();  x += "," + token;  }  mdt.add(x);  mdtc++;  addIntoMdt(ala.size()-1);  }    static void pass1Ala(String s) {  StringTokenizer st = new StringTokenizer(s, " ,", false);  String macro\_name = st.nextToken();  List<String> l = new ArrayList<>();  int index;  while(st.hasMoreTokens()) {  String x = st.nextToken();  if((index = x.indexOf("=")) != -1) {  x = x.substring(0, index);  }  l.add(x);  }  ala.add(l);  ala\_macro\_binding.put(macro\_name, ala\_macro\_binding.size());  }    static void addIntoMdt(int ala\_number) throws Exception {  String temp = new String();  String s = new String();  List l = ala.get(ala\_number);  boolean isFirst;  while(!s.equalsIgnoreCase("MEND")) {  isFirst = true;  s = input.readLine();  String line = new String();  StringTokenizer st = new StringTokenizer(s, " ,", false);  temp = st.nextToken();  for(int i=temp.length() ; i<12 ; i++) {  temp += " ";  }  line += temp;  while(st.hasMoreTokens()) {  temp = st.nextToken();  if(temp.startsWith("&")) {  int x = l.indexOf(temp);  temp = ",#" + x;  isFirst = false;  } else if(!isFirst) {  temp = "," + temp;  }  line += temp;  }  mdt.add(line);  mdtc++;  }  }    static void showAla(int pass) throws Exception {  PrintWriter out = new PrintWriter(new FileOutputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/out\_ala\_pass" + pass + ".txt"), true);  for(List l : ala) {  System.out.println(l);  out.println(l);  }  }    static void showMnt() throws Exception {  PrintWriter out = new PrintWriter(new FileOutputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/out\_mnt.txt"), true);  for(MntTuple l : mnt) {  System.out.println(l);  out.println(l);  }  }    static void showMdt() throws Exception {  PrintWriter out = new PrintWriter(new FileOutputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/out\_mdt.txt"), true);  for(String l : mdt) {  System.out.println(l);  out.println(l);  }  }    static List<String> pass2Ala(String s) {  StringTokenizer st = new StringTokenizer(s, " ", false);  int num\_tokens = st.countTokens();  String macro\_name = st.nextToken();  int ala\_no = ala\_macro\_binding.get(macro\_name);  List<String> l = ala.get(ala\_no);  int ctr = 0;  StringTokenizer st2 = null;  try {  st2 = new StringTokenizer(st.nextToken(), ",", false);  while(st2.hasMoreTokens()) {  l.set(ctr, st2.nextToken());  ctr++;  }  } catch(Exception e) {  // do nothing  }  if(ctr < num\_tokens) {  String s2 = mdt.get(mdtp);  StringTokenizer st3 = new StringTokenizer(s2, " ,", false);  String token = new String();  int index = 0;  while(st3.hasMoreTokens()) {  token = st3.nextToken();  if((index = token.indexOf("=")) != -1) {  try {  l.set(ctr++, token.substring(index+1, token.length()));  } catch(Exception e) {  // do nothing  }  }  }  }  ala.set(ala\_no, l);  return l;  }    static void initializeTables() {  mnt = new LinkedList<>();  mdt = new ArrayList<>();  ala = new LinkedList<>();  mntc = 0;  mdtc = 0;  ala\_macro\_binding = new HashMap<>();  }  } |

**OUTPUT**

|  |
| --- |
| 1. **Write a Java program for pass-II of a two-pass macro-processor. The output of assignment-3 (MNT, MDT and file without any macro definitions) should be input for this assignment.**   import java.util.\*;  import java.io.\*;  import static sposassignment.MacroProcessor.ala;  class MntTuple1 {  String name;  int index;    MntTuple1(String s, int i) {  name = s;  index = i;  }    public String toString() {  return("[" + name + ", " + index + "]");  }  }  class passtwoofthemacro {  static List<MntTuple1> mnt;  static List<String> mdt;  static int mntc;  static int mdtc;  static int mdtp;  static BufferedReader input;  static List<List <String>> ala;  static Map<String, Integer> ala\_macro\_binding;    public static void main(String args[]) throws Exception {  initializeTables();    System.out.println("\n===== PASS 2 =====\n");  pass2();  }    static void showAla(int pass) throws Exception {  PrintWriter out = new PrintWriter(new FileOutputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/out\_ala\_pass" + pass + ".txt"), true);  for(List l : ala) {  System.out.println(l);  out.println(l);  }  }    static void pass2() throws Exception {  input = new BufferedReader(new InputStreamReader(new FileInputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/output\_pass1.txt")));  PrintWriter output = new PrintWriter(new FileOutputStream("C:\\Users\\Lonkar\\Documents\\NetBeansProjects\\sposassignment\\macrotbl/output\_pass2.txt"), true);  String token = new String();  String s;  while((s = input.readLine()) != null) {  StringTokenizer st = new StringTokenizer(s, " ", false);  while(st.hasMoreTokens()) {  token = st.nextToken();  if(st.countTokens() > 2) {  token = st.nextToken();  }  MntTuple1 x = null;  for(MntTuple1 m : mnt) {  if(m.name.equalsIgnoreCase(token)) {  x = m;  break;  }  }  if(x != null) {  mdtp = x.index;  List<String> l = pass2Ala(s);  mdtp++;  String temp = new String();  while(!(temp = mdt.get(mdtp)).trim().equalsIgnoreCase("MEND")) {  String line = new String();  StringTokenizer st2 = new StringTokenizer(temp, " ,",false);  for(int i=0 ; i<12 ; i++) {  line += " ";  }  String opcode = st2.nextToken();  line += opcode;  for(int i=opcode.length() ; i<24 ; i++) {  line += " ";  }  line += st2.nextToken();  while(st2.hasMoreTokens()) {  String token2 = st2.nextToken();  int index;  if((index = token2.indexOf("#")) != -1) {  line += "," + l.get(Integer.parseInt(token2.substring(index+1,index+2)));  }  }  mdtp++;  output.println(line);  System.out.println(line);  }  break;  } else {  output.println(s);  System.out.println(s);  break;  }  }  }  System.out.println("\nALA:");  showAla(2);  }    static List<String> pass2Ala(String s) {  StringTokenizer st = new StringTokenizer(s, " ", false);  int num\_tokens = st.countTokens();  String macro\_name = st.nextToken();  int ala\_no = ala\_macro\_binding.get(macro\_name);  List<String> l = ala.get(ala\_no);  int ctr = 0;  StringTokenizer st2 = null;  try {  st2 = new StringTokenizer(st.nextToken(), ",", false);  while(st2.hasMoreTokens()) {  l.set(ctr, st2.nextToken());  ctr++;  }  } catch(Exception e) {  // do nothing  }  if(ctr < num\_tokens) {  String s2 = mdt.get(mdtp);  StringTokenizer st3 = new StringTokenizer(s2, " ,", false);  String token = new String();  int index = 0;  while(st3.hasMoreTokens()) {  token = st3.nextToken();  if((index = token.indexOf("=")) != -1) {  try {  l.set(ctr++, token.substring(index+1, token.length()));  } catch(Exception e) {  // do nothing  }  }  }  }  ala.set(ala\_no, l);  return l;  }    static void initializeTables() {  mnt = new LinkedList<>();  mdt = new ArrayList<>();  ala = new LinkedList<>();  mntc = 0;  mdtc = 0;  ala\_macro\_binding = new HashMap<>();  }  } |

**OUTPUT**

|  |
| --- |
| 1. **Write a program to create Dynamic Link Library for any mathematical operation and write an application program to test it. (Java Native Interface / Use VB or VC++).**   **Step:1 DLL file Code:**  Public Function powr(ByVal a As Integer, ByVal b As Integer) As Double  Dim ans As Integer  ans = 1  For i = 1 To b  ans = ans \* a  Next  powr = ans  End Function  Public Function oneby\_x(ByVal a As Double) As Double  oneby\_x = 1 / a  End Function  Public Function modd(ByVal a As Integer, ByVal b As Integer) As Double  modd = a Mod b  End Function  Public Function fact(ByVal a As Integer) As Integer  Dim f, i As Integer  f = 1  For i = 1 To a  f = f \* i  Next  fact = f  End Function  Public Function square(ByVal a As Double) As Double  square = a \* a  End Function  Public Function logg(ByVal a As Double) As Double  logg = Log(a) / Log(10)  End Function  Public Function tann(ByVal a As Double) As Double  Dim c As Double  c = a \* 3.14 / 180  tann = Tan(c)  End Function  Public Function coss(ByVal a As Double) As Double  Dim c As Double  c = a \* 3.14 / 180  coss = Cos(c)  End Function  Public Function sinn(ByVal a As Double) As Double  Dim c As Double  c = a \* 3.14 / 180  sinn = Sin(c)  End Function  Public Function root(ByVal a As Integer) As Integer  root = Sqr(a)  End Function  Public Function division(ByVal a As Double, ByVal b As Double) As Double  division = a / b  End Function  Public Function multiplication(ByVal a As Double, ByVal b As Double) As Double  multiplication = a \* b  End Function  Public Function subtraction(ByVal a As Double, ByVal b As Double) As Double  subtraction = a - b  End Function  Public Function addition(ByVal a As Double, ByVal b As Double) As Double  addition = a + b  End Function  **Step 2:Standard EXE project code:**  Dim a, b, opr As Integer  Dim obj As New operations  Dim str As String  Private Sub cmd\_ON\_Click()  Text1.Text = "0"  End Sub  Private Sub cmd1\_Click()  a = Val(Text1.Text)  If (a = 0) Then  MsgBox "Divide by zero", vbInformation  Text1.Text = ""  Else  Text1.Text = obj.oneby\_x(a)  End If  End Sub  Private Sub cmdadd\_Click()  a = Val(Text1.Text)  Text1.Text = ""  opr = 1  End Sub  Private Sub cmdback\_Click()  If (Text1.Text <> "") Then  str = Text1.Text  Text1.Text = Left(str, (Len(str) - 1))  End If  End Sub  Private Sub cmdclr\_Click()  Text1.Text = "0"  End Sub  Private Sub cmdcos\_Click()  a = Val(Text1.Text)  Text1.Text = obj.coss(a)  End Sub  Private Sub cmddiv\_Click()  a = Val(Text1.Text)  Text1.Text = ""  opr = 4  End Sub  Private Sub cmdequal\_Click()  b = Val(Text1.Text)  If opr = 1 Then  Text1.Text = obj.addition(a, b)  End If  If opr = 2 Then  Text1.Text = obj.subtraction(a, b)  End If  If opr = 3 Then  Text1.Text = obj.multiplication(a, b)  End If  If opr = 4 Then  If (b = 0) Then  MsgBox "Divde by zero", vbInformation  Text1.Text = ""  Else  Text1.Text = obj.division(a, b)  End If  End If  If opr = 5 Then  Text1.Text = obj.modd(a, b)  End If  If opr = 6 Then  Text1.Text = obj.powr(a, b)  End If  End Sub  Private Sub cmdfact\_Click()  a = Val(Text1.Text)  Text1.Text = obj.fact(a)  End Sub  Private Sub cmdlog\_Click()  a = Val(Text1.Text)  Text1.Text = obj.logg(a)  End Sub  Private Sub cmdminus\_Click()  a = Val(Text1.Text)  Text1.Text = ""  opr = 2  End Sub  Private Sub cmdmod\_Click()  a = Val(Text1.Text)  Text1.Text = ""  opr = 5  End Sub  Private Sub cmdmul\_Click()  a = Val(Text1.Text)  Text1.Text = ""  opr = 3  End Sub  Private Sub cmdoff\_Click()  Text1.Text = ""  End Sub  Private Sub cmdpowr\_Click()  a = Val(Text1.Text)  opr = 6  Text1.Text = ""  End Sub  Private Sub cmdroot\_Click()  a = Val(Text1.Text)  Text1.Text = obj.root(a)  End Sub  Private Sub cmdsin\_Click()  a = Val(Text1.Text)  Text1.Text = obj.sinn(a)  End Sub  Private Sub cmdsqr\_Click()  a = Val(Text1.Text)  Text1.Text = obj.square(a)  End Sub  Private Sub cmdtan\_Click()  a = Val(Text1.Text)  Text1.Text = obj.tann(a)  End Sub  Private Sub Command1\_Click(Index As Integer)  Text1.Text = Text1.Text + Command1(Index).Caption  End Sub  Private Sub Command9\_Click()  Text1.Text = "0"  End Sub  Private Sub Command2\_Click()  Text1.Text = Text1.Text + "."  End Sub  Private Sub Form\_Load()  Text1.Text = "0"  End Sub |

**OUTPUT**

|  |
| --- |
| 1. **Write a program using Lex specifications to implement lexical analysis phase of compiler to generate tokens of subset of ‘Java’ program.** |

**OUTPUT**

|  |
| --- |
| 1. **Write a program using Lex specifications to implement lexical analysis phase of compiler to count no. of words, lines and characters of given input file.** |

**OUTPUT**

|  |
| --- |
| 1. **Write a program using YACC specifications to implement syntax analysis phase of compiler to validate type and syntax of variable declaration in Java.** |

**OUTPUT**

|  |
| --- |
| 1. **Write a program using YACC specifications to implement syntax analysis phase of compiler to recognize simple and compound sentences given in input file.**   **OUTPUT**   1. **Write a Java program (using OOP features) to implement following scheduling algorithms:**   **OUTPUT**   1. **FCFS , SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive)**   **Step 1:Open NetBeans and copy paste the code.(FCFS)**  import java.io.\*;  class FCFS  {  public static void main(String args[]) throws Exception  {  int n,bt[],wt[],tat[];  float awt=0;  InputStreamReader isr=new InputStreamReader(System.in);  BufferedReader br=new BufferedReader(isr);  System.out.println("Enter no of process");  n=Integer.parseInt(br.readLine());  wt=new int[n];  tat=new int[n];  bt=new int[n];  System.out.println("Enter Burst time for each process\n");  for(int i=0;i<n;i++)  {  System.out.println("Process["+(i+1)+"]");  bt[i]=Integer.parseInt(br.readLine());  }  System.out.println("\n");  wt[0]=0;  for(int i=1;i<n;i++)  {  wt[i]=wt[i-1]+bt[i-1];  awt=awt+wt[i];  }  for(int i=0;i<n;i++)  {  tat[i]=wt[i]+bt[i];  }  System.out.println(" PROCESS\t\tBURST-TIME\tWAITING-TIME\tTURN AROUND-TIME\n");  for(int i=0;i<n;i++)  {  System.out.println(" "+ i + "\t\t"+bt[i]+"\t"+wt[i]+"\t"+tat[i]);  }  awt=awt/n;  System.out.println("\n");  System.out.println("Avg waiting time="+awt+"\n");  }  }  **OUTPUT**  **Step 2:Open NetBeans and copy paste the code.(Priority)**  import java.util.Scanner;  public class TestClass {    public static void main(String args[]) {  Scanner s = new Scanner(System.in);  int x,n,p[],pp[],bt[],wt[],tat[],awt,atat,i;  p = new int[10];  pp = new int[10];  bt = new int[10];  w = new int[10];  t = new int[10];  //n is number of process  //p is process  //pp is process priority  //bt is process burst time  //wt is wait time  // tat is turnaround time  //awt is average waiting time  //atat is average turnaround time  System.out.print("Enter the number of process : ");  n = s.nextInt();  System.out.print("\n\t Enter burst time : time priorities \n");  for(i=0;i<n;i++)  {  System.out.print("\nProcess["+(i+1)+"]:");  bt[i] = s.nextInt();  pp[i] = s.nextInt();  p[i]=i+1;  }  //sorting on the basis of priority  for(i=0;i<n-1;i++)  {  for(int j=i+1;j<n;j++)  {  if(pp[i]<pp[j])  {  x=pp[i]; /\*  pp[i]=pp[j];  pp[j]=x; swapping of priorities of processes using x as temp \*/  x=bt[i]; /\*  bt[i]=bt[j];  bt[j]=x; swapping of burst time using x as temp \*/  x=p[i]; /\*  p[i]=p[j];  p[j]=x; swapping of process nos. using x as temp \*/  }  }  }  wt[0]=0; //waiting time of first process is 0  awt=0;  tat[0]=bt[0]; //turn around time of 1st process is burst time of it  int atat=tat[0];  for(i=1;i<n;i++)  {  wt[i]=tat[i-1];  awt+=wt[i];  tat[i]=wt[i]+bt[i];  atat+=tat[i];  }  //Displaying the process  System.out.print("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time Priority \n");  for(i=0;i<n;i++)  System.out.print("\n "+p[i]+"\t\t "+bt[i]+"\t\t "+wt[i]+"\t\t "+tat[i]+"\t\t "+pp[i]+"\n");  awt/=n;  atat/=n;  System.out.print("\n Average Wait Time : "+awt);  System.out.print("\n Average Turn Around Time : "+atat);  }  }  **OUTPUT**  **Step 3:Open NetBeans and copy paste the code.(Round Robin)**  import java.util.Scanner;  public class TestClass  {    public static void main(String args[])  {  Scanner s = new Scanner(System.in);  int wt[],bt[],rt[],num,quantum,total;  wt = new int[10];  bt = new int[10];  rt = new int[10];  tat = new int[10];  System.out.print("Enter number of processes(MAX 10): ");  num = s.nextInt();  System.out.print("Enter burst time");  for(int i=0;i<num;i++)  {  System.out.print("\nP["+(i+1)+"]: ");  bt[i] = s.nextInt();  rt[i] = btime[i];  wt[i]=0;  }  System.out.print("\n\nEnter quantum: ");  quantum = s.nextInt();  int rp = num;  int i=0;  int time=0;  System.out.print("0");  wtime[0]=0;  while(rp!=0) //while no. of processes are not 0  {  if(rtime[i]>quantum) //if burst time is > quantum  {  rtime[i]=rtime[i]-quantum; //burst time of that process updated  System.out.print(" | P["+(i+1)+"] | ");  time+=quantum; //time updated  System.out.print(time);  }  else if(rtime[i]<=quantum && rtime[i]>0) //if burst time is > 0 but less than quantum  {  time+=rtime[i]; //time updated(time + burst time)  rtime[i]=rtime[i]-rtime[i]; //burst time of that process is updated(it will be 0)  System.out.print(" | P["+(i+1)+"] | ");  ` rp--; // no. of processes updated(since current process completed its execution)  System.out.print(time);  }    i++; //go to next process  if(i==num) //if last process then goes to first process(cycle repeats)  {  i=0;  }    }      }  }  **OUTPUT**  **Step 3:Open NetBeans and copy paste the code.(SJF)**  import java.util.\*;    public class SRTF {  public static void main (String args[])  {  Scanner sc=new Scanner(System.in);  System.out.println ("enter no of process:");  int n= sc.nextInt();  int p[] = new int[n]; // it takes pid of process  int at[] = new int[n]; // at means arrival time  int bt[] = new int[n]; // bt means burst time  int ct[] = new int[n]; // ct means complete time  int ta[] = new int[n]; // ta means turn around time  int wt[] = new int[n]; // wt means waiting time  int f[] = new int[n]; // f means it is flag it checks process is completed or not  int k[]= new int[n]; // it is also stores brust time  int i, st=0, tot=0;  float awt=0, ata=0;    for (i=0;i<n;i++)  {  p[i]= i+1;  System.out.println ("enter process " +(i+1)+ " arrival time:");  at[i]= sc.nextInt();  System.out.println("enter process " +(i+1)+ " burst time:");  bt[i]= sc.nextInt();  k[i]= bt[i];  f[i]= 0;  }  while(true){  int min=99,c=n;  if (tot==n) //total  break;    for ( i=0;i<n;i++)  {  if ((at[i]<=st) && (f[i]==0) && (bt[i]<min)) //if arrival time less than equal to st(first time 0 then 1 & so on & process not completed & burst time < min(99 assumed)  {  min=bt[i]; //burst time of that process saved in min  c=i; // index of that process is stored in c  }  }  if (c==n) //if no process's arrival time is 0  st++; //st incremented  else  {  bt[c]--; //burst time decremented by 1(since it executed for 1 msec)  st++; // st incremented  if (bt[c]==0) //it burst time of that process is 0  {  ct[c]= st;  f[c]=1; //completed  tot++; //total updated  }  }  }  for(i=0;i<n;i++)  {  ta[i] = ct[i] - at[i];  wt[i] = ta[i] - k[i];  awt+= wt[i]; //total waiting time calculated  ata+= ta[i]; //total turn around time calculated  }  System.out.println("p arrival burst complete turn waiting");  for(i=0;i<n;i++)  {  System.out.println(p[i] +"\t"+ at[i]+"\t"+ k[i] +"\t"+ ct[i] +"\t"+ ta[i] +"\t"+ wt[i]);  }  System.out.println("\naverage tat is "+ (float)(avgta/n));  System.out.println("average wt is "+ (float)(avgwt/n));  sc.close();  }   1. **Write a Java program to implement Banker’s Algorithm**   **Step 1:Open NetBeans and copy paste the code.**  import java.util.\*;  public class Bankers {  public static void main(String[] args) {  Scanner sc=new Scanner(System.in);  System.out.println("Enter number Of Process:");  int n = sc.nextInt();  System.out.println("Enter number Of Resources:");  int m = sc.nextInt();  System.out.println("\n============================== ");  int available[] = new int[m];  int max[][] = new int[n][m];  int allocation[][] = new int[n][m];  int need[][] = new int[n][m];  String sequence = "";    //Allocoation  for(int i = 0; i < n; i++)  {  for(int j = 0; j < m; j++)  {  System.out.println("Allocation P" + (i) + " for R " + (j) + ":");  allocation[i][j] = sc.nextInt();    //calcultaing available  available[j]+=allocation[i][j];  }  }  System.out.println("\n============================== ");  //Displaying Available  for(int i=0;i<m;i++) {  System.out.println("Available for Resource R"+i);  System.out.println(available[i]+" ");  }  System.out.println("\n============================== ");  //Max and Needs  for(int i = 0; i < n; i++)  {  for(int j = 0; j < m; j++)  {  System.out.println("Max P" + (i) + " for R " + (j) + ":");  max[i][j] = sc.nextInt();  need[i][j] = max[i][j] - allocation[i][j];  }  }  System.out.println("\n============================== ");  int work[] = available;  boolean finish[] = new boolean[n];  for(int i = 0; i < n; i++)  {  finish[i] = false;  }    boolean check = true;  while(check)  {  check = false;  for(int i = 0; i < n; i++)  {  if(!finish[i])  {  int j;  for(j = 0; j < m; j++)  {  if(need[i][j] > work[j])  {  break;  }  }  if(j == m)  {  for(j=0; j < m; j++)  {  work[j] = work[j] + allocation[i][j];  }  finish[i] = true;  check = true;  sequence += i + ", ";  }  }  }  }  int i;  for(i = 0; i < n; i++)  {  if(!finish[i])  break;  }    if(i==n)  {  System.out.println("Safe And Sequence is:"+sequence);  }  else  {  System.out.println("Unsafe");  }    }  }  **OUTPUT**   1. **Implement UNIX system calls like ps, fork, join, exec family, and wait for process management (use shell script/ Java/ C programming).**   #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <unistd.h>  #include <sys/wait.h>  int main( int argc, char \*argv[], char \*env[] )  {  pid\_t my\_pid, parent\_pid, child\_pid;  int status;  /\* get and print my pid and my parent's pid. \*/  my\_pid = getpid(); parent\_pid = getppid();  printf("\n Parent: my pid is %d\n\n", my\_pid);  printf("Parent: my parent's pid is %d\n\n", parent\_pid);  /\* print error message if fork() fails \*/  if((child\_pid = fork()) < 0 )  {  perror("fork failure");  exit(1);  }  /\* fork() == 0 for child process \*/  if(child\_pid == 0)  { printf("\nChild: I am a new-born process!\n\n");  my\_pid = getpid(); parent\_pid = getppid();  printf("Child: my pid is: %d\n\n", my\_pid);  printf("Child: my parent's pid is: %d\n\n", parent\_pid);  printf("Child: I will sleep 3 seconds and then execute - date - command \n\n");  sleep(3);  printf("Child: Now, I woke up and am executing date command \n\n");  execl("/bin/date", "date", 0, 0);  perror("execl() failure!\n\n");  printf("This print is after execl() and should not have been executed if execl were successful! \n\n");  \_exit(1);  }  /\*  \* parent process  \*/  else  {  printf("\nParent: I created a child process.\n\n");  printf("Parent: my child's pid is: %d\n\n", child\_pid);  system("ps -acefl | grep ercal"); printf("\n \n");  wait(&status); /\* can use wait(NULL) since exit status  from child is not used. \*/  printf("\n Parent: my child is dead. I am going to leave.\n \n ");  }  return 0;  }  **OUTPUT**   1. **Study assignment on process scheduling algorithms in Android and Tizen.**   **OUTPUT** |
| 1. **Write a Java Program (using OOP features) to implement paging simulation using** 2. **Least Recently Used (LRU)** import java.util.Scanner;     public class LRU {    public static int min(int counter[],int nFrames) //function to find least recently used page  {  int minimum = counter[0]; //count of 1st page is considered as minum  int pos = 0;  for(int i=0;i<nFrames;i++) //finding page which have less count  {  if(minimum > counter[i])  pos = i;  }  return pos; //position of that page is returned  }    public static void main(String[] args) {  // TO DO code application logic here  Scanner s = new Scanner(System.in);  int n,recent = 0,pageFault = 0,nFrames;    System.out.print("Enter the number of pages: ");  n = s.nextInt();  int pageString[] = new int[n];  System.out.print("Enter the page reference string: ");  for(int i=0;i<n;i++)  pageString[i]=s.nextInt();    System.out.print("\nEnter the number of frames: ");  nFrames = s.nextInt();  int frames[] = new int[nFrames];  int counter[] = new int[nFrames];    for(int i=0;i<nFrames;i++)  { frames[i] = 0;  counter[i] = 0; //here 0 referes an empty space in frame  }    for(int i=0;i<n;i++)  {  int flag =0;  for(int j=0;j<nFrames;j++)  {  if(frames[j] == pageString[i]) //checking whether page is already present in frame  {  flag=1; // flag is set  counter[j] = recent++; //counter holds which frame is recently used,  //recently used page in frame will have a bigger number  //and least recently used page in frame will have a lower number  break;  }  }    if(flag == 0) // page is not present in frame  {  for(int j=0;j<nFrames;j++)  {  if(frames[j] == 0) //space is available in frame  { frames[j] = pageString[i];  counter[j] = recent++;  flag=1;  pageFault++; // page fault will occur  break;  }    }  }    if(flag == 0) //space is not present in frame  {  int PositionToreplace = min(counter,nFrames);  frames[PositionToreplace] = pageString[i];  counter[PositionToreplace] = recent++;  pageFault++;  }    //print frames  System.out.println();  for(int j=0;j<nFrames;j++)  {  System.out.print(frames[j]+" ");  }    }    System.out.print("\nPage Fault: "+pageFault);    }    }  **OUTPUT**  **2. Optimal algorithm**  import java.util.Scanner;  public class Optimal {  public static void main(String[] args)  {  int n,m,i,j,k,mn,flag=0,pgfault=0;  Scanner sc=new Scanner(System.in);  System.out.println("Enter no of pages");  n=sc.nextInt();  System.out.println("enter page nos");  int a[]=new int[n];  for(i=0;i<n;i++)  a[i]=sc.nextInt();  System.out.println("Enter no of frames");  m=sc.nextInt();  int frm[]=new int[m];  int whn[]=new int[m];  for(i=0;i<m;i++)  {  frm[i]=a[i];  whn[i]=0;  }  for(;i<n;i++)  {  for(j=0;j<m;j++)  {  if(frm[j]==a[i])  {  flag=1;  break;  }  else  flag=0;  }  if(flag==0)  {  for(j=0;j<m;j++)  {  for(k=i+1;k<n;k++)  {  if(frm[j]==a[k])  whn[j]=k;  else  whn[j]=0;  }  }  mn=ltr(whn,m);  frm[mn]=a[i];  pgfault++;  }  }  pgfault=pgfault+m;  System.out.println("Optimal algorithm");  System.out.println("page fault is:"+pgfault);  System.out.println("page hits are:"+(n-pgfault));  }  static int ltr(int whn[],int m)  {  int i,mn=0;  for(i=1;i<m;i++)  if((whn[i] > whn[i-1]) && (whn[i]!=0))  mn=i;  return mn;  }  }  **OUTPUT** |