STABLE MATCHING

BY:

Anurag Pareek 031262
Madhur Chawla 031417

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

May 2007

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, **"Stable Matching"** submitted by Anurag Pareek and Madhur Chawla in partial fulfillment for the award of degree of Bachelor of Technology in 2007 of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Nitin
21st May 2007

Nitin
Sr. Lecturer
Department Of Computer Science & Engineering

Brig. S.P.Ghrera
HOD, Department of CSE & IT

Acknowledgement

We would like to express our deep sense of gratitude and heartiest thanks to our teacher ***Mr. Nitin,*** Senior Lecturer, Jaypee University of Information Technology for guiding us throughout this project and providing us each and every resource required to make this project a success.

We would also like to thank Dr. Kenneth J. Goldman; Associate Professor, Department of Computer Science and Engineering, Washington University for providing us guidelines on Gale Shapley algorithm basic developed in java.

Furthermore, we would like to thank all the friends who helped us directly or indirectly in project development inclusive of typical programming involved in this project.

Lastly, we would like to show our high gratitude to the entire faculty for showing us different methodologies used in our project.

Anurag Pareek (031262)

Madhur Chawla (031417)

Table of Contents

LIST OF FIGURES

Abstract

"Computer Networks" a field which is known as a pillar for the spread of technology. Day by Day things are going bigger and same is the complexity. So a question arises what is the reliability of the network. How will it work and this issue is so prominent that it make the father of programming Knuth to think upon it however it remain unanswered at that time . The task was to find a formulation to find the best matching between two categories to attain stability.

Stable Marriage or Stable Matching is the name for which we are talking about. In an instance of size n of the stable marriage problem, each of n men and n women ranks the members of the opposite sex in order of preference. A stable matching is a complete matching of men and women such that no man and woman who are not partners both prefer each other to their actual partners under the matching. It is well known that at least one stable matching exists for every stable marriage instance. However, the classical Gale-Shapley algorithm produces a marriage that greatly favors the men at the expense of the women, or vice versa. The problem arises of finding a stable matching that is optimal under some more equitable or egalitarian criterion of optimality. This problem was posed by Knuth and has remained unsolved for some time. Here, the objective of maximizing the average (or, equivalently, the total) "satisfaction" of all people is used. This objective is achieved when a person's satisfaction is measured by the position of his/her partner in his/her preference list.

A java applet to find matching based on Gale Shapley and a C code based on gale Shapley and Irving's rotational method with the introduction to interconnection multistage network.

Chapter 1: Computer Network

1.1 Network

A computer network is multiple computers connected together using a telecommunication system for the purpose of communicating and sharing resources.
A computer network may be described as the interconnection of two or more computers that may share files and folders, applications, or resources like printers, scanners, webcams etc. Internet is also a type of computer network which connects all the computers of the world having Internet facility on them.

1.2 Advantages

- Peripherals (e.g. printers) can be shared
- Software can be shared
- Data can be shared
- Computers may communicate
- Security - user access may be restricted if needed

1.3 Disadvantages of Networks

- One server breaking down may affect a number of computers
- Vulnerable to hackers and viruses
- Cabling and installation may be expensive
- A network manager may need to be employed to run the network

Chapter 2: Metrics for Networks

Metrics provide a framework to compare and evaluate networks. Some of the metrics are listed below with networking algorithms:-

2.1 Network Connectivity

Network nodes and communication links sometimes fail and must be removed from service for repair. When components do fail the network should continue to function with reduced capacity.

Network connectivity measures the resiliency of a network and its ability to continue operation despite disabled components i.e. connectivity is the minimum number of nodes or links that must fail to partition the network into two or more disjoint networks

The larger the connectivity for a network the better the network is able to cope with failures.

2.2 Network Diameter

The diameter of a network is the maximum internodes distance i.e. it is the maximum number of links that must be traversed to send a message to any node along a shortest path.

The lower the diameter of a network the shorter the time to send a message from one node to the node farthest away from it.

2.3 Narrowness

This is a measure of congestion in a network and is calculated as follows:

Partition the network into two groups of processors A and B where the number of processors in each group is Na and Nb and assume Nb < = Na. Now count the number of interconnections between A and B call this I. Find the maximum value of Nb / I for all partitioning of the network. This is the narrowness of the network.

The idea is that if the narrowness is high (Nb > I) then if the group B processors want to send messages to group A congestion in the network will be high (since there are fewer links than processors)

2.4 Network Expansion Increments

A network should be expandable i.e. it should be possible to create larger and more powerful multi-computer systems by simply adding more nodes to the network.

For reasons of cost it is better to have the option of small increments since this allows you to upgrade your network to the size you require (i.e. flexibility) within a particular budget.

E.g. an 8 node linear array can be expanded in increments of 1 node but a 3 dimensional hypercube can be expanded only by adding another 3D hypercube. (i.e. 8 nodes)

2.5 Networking Algorithms

- Shortest paths.
- Maximum and Minimum cost flow.
- Matching.
- Stable Marriage.

Chapter 3: Stable Marriage

In this section, we will first briefly describe the Gale-Shapley algorithm.
After that we will explain Irving Theorem

Networking Algorithms

- Shortest paths.
- Maximum and Minimum cost flow.
- Matching.
- Stable Marriage.

3.1 The Gale-Shapley Algorithm

We review the well-known algorithm of Gale and Shapley, not only because of its general importance but because the private stable matching protocols presented later are, in fact, simulations of variants of the Gale-Shapley algorithm. The Gale-Shapley algorithm considers a series of proposals made by men, round-by-round. Whenever a proposal is accepted, the couple is considered engaged. If a man is not engaged, he is considered free. The algorithm proceeds as follows. If there are any free men, select one at random (call him A). A proposes to the woman he ranks highest among those to whom he has not yet proposed (call her B). If B is free, she accepts and the pair is considered engaged. If B is engaged to some A0 and she ranks A0 ahead of A, then B and A0 remain engaged and A remains free. If B is engaged to A0 and she ranks A0 below A, then B and A become engaged and A0 becomes free. After $O(n^2)$ proposals, all participants will be engaged and we will have found a stable marriage. In fact, the marriage we find is men-optimal. Due to symmetry, it is clear we could run the algorithm to find a marriage that is women-optimal.

3.2 The Irving Algorithm

The classical algorithm normally yields what is called the male optimal solution, with the property that every man has the best partner that he can have in any stable matching. If applied with the roles of men and women interchanged, the algorithm will yield the female optimal solution, which similarly favors the women. However, it turns out that the achievement of best possible partners by the members of one sex results in the members of the opposite sex having their worst possible partners, so that the question has been raised, for example by Knuth, and others as to whether there exists an efficient algorithm to find the optimal stable marriage under a more equitable measure of optimality. To find a stable marriage that maximizes the total satisfaction of all the men and women, where a person's satisfaction is measured by the position of his/her partner in his/her preference list. An instance of the stable marriage problem may be specified by the male and female ranking matrices. Relative to arbitrary but fixed numberings of males and females, these are defined by

$mr(i, k) = j$ if woman k is the j^{th} choice of man i,

$wr(i, k) = j$ if man k is the j^{th} choice of woman i.

The problem of how to find a stable marriage maximizing total satisfaction. Suppose that, for a given stable marriage instance,

$S = \{(m_1, w_1) \ldots (m_n, w_n)\}$

And we say that a stable matching S is optimal if it has minimum possible value c(S). It is clear that a stable matching that is optimal in this sense maximizes the Total (or average) satisfaction of the 2n individuals. Such an optimal stable matching can certainly be found by generating all stable matching and comparing their values.

Chapter 4: Source Code

4.1 C Source Code

```
/*  Program Based on Gale-Shapley Algorithm and Gusfield & Irving*/

#include <stdio.h>
#include<conio.h>
# include<stdlib.h>

typedef struct pair pairT;      /* New User Defined Data Type */

struct pair {
  int male,maleRank,female,femaleRank;
  pairT *nextFemale,*prevFemale,*nextMale,*prevMale;   /* pointer for ranking of
next/previous male or female */
};

pairT pair[40][40];

pairT *maleHeader[40],*femaleHeader[40];

int n;

int maleInput[40][40],femaleInput[40][40];

int maleOptimal[40],femaleOptimal[40];

void saveInput()      // Function to take the input from the user
{
```

```
int i,j;

printf("\nProgram for Stable Marriage Algorithm as a part of ");
printf("\nFinal Year Project BTech 8th Sem CSE");
printf("\n==========================================================
=====");
printf("\nProject Incharge :- Sr. Lect Mr . Nitin");
printf("\nBy :- Anurag Pareek (031262),Madhur Chawla(031417)");
printf("\n==========================================================
=====");
printf("\nFor Any help and suggestion please contact ");
printf("\ndelnitin@yahoo.co.in,pareek.anurag@gmail.com");
printf("\n==========================================================
=====");
printf("\nProgram Based on Gale-Shapley And Gusfield & Irving Algorithm");
printf("\n==========================================================
=====");
enter:
printf("\nPlease Enter Number of Persons less then 40 : ");
scanf("%d",&n);
if(n >=40)
{  printf("\nPlease Enter Less Than 40");
   goto enter;
}
printf("\n==========================================================
=====");
printf("\nEnter Male Preferences : ");
printf("\n==========================================================
=====");
for (i=1;i<=n;i++)
{  for (j=1;j<=n;j++)
```

```
        { printf("\nEnter Preference List For Male %d : ",i);
          mlist:
          scanf("%d",&maleInput[i][j]);
          if(maleInput[i][j] > n)
          {  printf("Prefrence List Should be Less than %d . Enter again :",n);
             goto mlist;
          }

        }
    }

    printf("\n==========================================================
    =====");
    printf("\nEnter Female Preferences : ");
    printf("\n==========================================================
    =====");
    for (i=1;i<=n;i++)
    {  for (j=1;j<=n;j++)
        { printf("\nEnter Preference List For Female %d : ",i);
          flist:
          scanf("%d",&femaleInput[i][j]);
          if(maleInput[i][j] > n)
          {   printf("Prefrence List Should be Less than %d . Enter again",n);
              goto flist;
          }
        }
    }

    }

    void readInput() // add the given preference list on the Linked List
```

```
{
int i,j,male,female,rank;
pairT *ptr;

for (i=1;i<=n;i++) /*initialization*/
{
 maleHeader[i]= &pair[i][0];  /*initialize male header*/
 ptr=maleHeader[i];
 ptr->male=i;
 ptr->maleRank=0;
 ptr->female=0;
 ptr->femaleRank=0;
 ptr->nextFemale=ptr;
 ptr->prevFemale=ptr;
 ptr->nextMale=NULL;
 ptr->prevMale=NULL;
 femaleHeader[i]= &pair[0][i];  /*initialize female header*/
 ptr=femaleHeader[i];
 ptr->male=0;
 ptr->maleRank=0;
 ptr->female=i;
 ptr->femaleRank=0;
 ptr->nextMale=ptr;
 ptr->prevMale=ptr;
 ptr->nextFemale=NULL;
 ptr->prevFemale=NULL;

 for (j=1;j<=n;j++)
 {
        ptr= &pair[i][j];  /*initialize a pair node*/
        ptr->male=i;
```

```
            ptr->maleRank=0;
            ptr->female=j;
            ptr->femaleRank=0;
            ptr->nextFemale=NULL;
            ptr->prevFemale=NULL;
            ptr->nextMale=NULL;
            ptr->prevMale=NULL;
    }
}

for (male=1;male<=n;male++)  /*read men's preference lists*/
 for (rank=1;rank<=n;rank++)
 {
        female=maleInput[male][rank];
        ptr= &pair[male][female];
        if (ptr->maleRank!=0)
        {
                printf("Male list has repeat\n");
                exit(0);
        }
        ptr->maleRank=rank;
        ptr->nextFemale=maleHeader[male];
        ptr->prevFemale=maleHeader[male]->prevFemale;
        maleHeader[male]->prevFemale->nextFemale=ptr;
        maleHeader[male]->prevFemale=ptr;
}

for (female=1;female<=n;female++)  /*read females' preference lists*/
 for (rank=1;rank<=n;rank++)
 {
        male=femaleInput[female][rank];
```

```
            ptr= &pair[male][female];
            if (ptr->femaleRank!=0)
            {
                  printf("Female list has repeat\n");
                  exit(0);
            }
            ptr->femaleRank=rank;
            ptr->nextMale=femaleHeader[female];
            ptr->prevMale=femaleHeader[female]->prevMale;
            femaleHeader[female]->prevMale->nextMale=ptr;
            femaleHeader[female]->prevMale=ptr;
      }
}

void readInputRolesReversed()
{
int i,j,male,female,rank;
pairT *ptr;

for (i=1;i<=n;i++) /*initialization*/
{
 maleHeader[i]= &pair[i][0];  /*initialize male header*/
 ptr=maleHeader[i];
 ptr->male=i;
 ptr->maleRank=0;
 ptr->female=0;
 ptr->femaleRank=0;
 ptr->nextFemale=ptr;
 ptr->prevFemale=ptr;
 ptr->nextMale=NULL;
 ptr->prevMale=NULL;
```

```
    femaleHeader[i]= &pair[0][i];  /*initialize female header*/
    ptr=femaleHeader[i];
    ptr->male=0;
    ptr->maleRank=0;
    ptr->female=i;
    ptr->femaleRank=0;
    ptr->nextMale=ptr;
    ptr->prevMale=ptr;
    ptr->nextFemale=NULL;
    ptr->prevFemale=NULL;
    for (j=1;j<=n;j++)
    {
        ptr= &pair[i][j];  /*initialize a pair node*/
        ptr->male=i;
        ptr->maleRank=0;
        ptr->female=j;
        ptr->femaleRank=0;
        ptr->nextFemale=NULL;
        ptr->prevFemale=NULL;
        ptr->nextMale=NULL;
        ptr->prevMale=NULL;
    }
}

for (male=1;male<=n;male++)  /*read men's preference lists*/
 for (rank=1;rank<=n;rank++)
 {
        female=femaleInput[male][rank];  /*Roles reversed*/
        ptr= &pair[male][female];
        if (ptr->maleRank!=0)
        {
```

```
                printf("Male list has repeat\n");
                abort();
            }
            ptr->maleRank=rank;
            ptr->nextFemale=maleHeader[male];
            ptr->prevFemale=maleHeader[male]->prevFemale;
            maleHeader[male]->prevFemale->nextFemale=ptr;
            maleHeader[male]->prevFemale=ptr;
    }

    for (female=1;female<=n;female++)  /*read females' preference lists*/
     for (rank=1;rank<=n;rank++)
     {
            male=maleInput[female][rank]; /*Roles reversed*/
            ptr= &pair[male][female];
            if (ptr->femaleRank!=0)
            {
                printf("Female list has repeat\n");
                abort();
            }
           ptr->femaleRank=rank;
           ptr->nextMale=femaleHeader[female];
           ptr->prevMale=femaleHeader[female]->prevMale;
           femaleHeader[female]->prevMale->nextMale=ptr;
           femaleHeader[female]->prevMale=ptr;
     }
    }

    void printMatching()
    {
    int i;
```

```
/*Result is the first remaining node on male lists*/
for (i=1;i<=n;i++)
  printf("%d %d\n",i,maleHeader[i]->nextFemale->female);
}

void printPreferenceLists()
{
int i;
pairT *ptr;
printf("\n==========================================================
======");
printf("\nMale preference lists is :");
printf("\n==========================================================
======\n");
for (i=1;i<=n;i++)
{
 printf("%d: ",i);
 ptr=maleHeader[i]->nextFemale;
 while (ptr->female!=0)
 {
        printf("%d ",ptr->female);
        ptr=ptr->nextFemale;
 }
 printf("\n");
}
printf("\n==========================================================
======\n");
printf("\nFemale preference lists is :");
printf("\n==========================================================
======\n");
```

```
    for (i=1;i<=n;i++)
    {
     printf("%d: ",i);
     ptr=femaleHeader[i]->nextMale;
     while (ptr->male!=0)
     {
          printf("%d ",ptr->male);
          if (ptr->nextMale->prevMale!=ptr)
               printf("Link Error\n");
          ptr=ptr->nextMale;
     }
    printf("\n");
  }
  }

  void galeShapley()
  {
  /*Finds male-optimal solution and prunes lists*/

  int stack[40],sp,i,maleProposer,femaleEngaged[40];
  int female;
  pairT *ptr;

 sp= -1;
 for (i=1;i<=n;i++)
 {
  sp++;
  stack[sp]=i;
  femaleEngaged[i]=0;
 }
```

```
/*Gale-Shapley Algorithm*/
while (sp!=(-1))
{
 maleProposer=stack[sp];
 sp--;
 female=maleHeader[maleProposer]->nextFemale->female;
 printf("Male %d proposes to female %d\n",maleProposer,female);
 getch();
 if (femaleEngaged[female])
 {
        /*Female changes engagement, since list deletions*/
        /*guarantee that only better pairs remain for a female*/
        printf("\n\nFemale %d breaks engagement with male %d\n",
                female,femaleHeader[female]->prevMale->male);
        sp++;
        stack[sp]=femaleHeader[female]->prevMale->male;
 }
 else
 { getch();
        printf("\n\nFirst engagement for female %d\n",female);
        femaleEngaged[female]=1;
 }
ptr=femaleHeader[female]->prevMale;
while (ptr->male!=maleProposer)
{
        /*Delete less preferable males for female out of men's lists*/
        printf("\n\nDelete male=%d female=%d\n",ptr->male,female);
        ptr->prevFemale->nextFemale=ptr->nextFemale;
        ptr->nextFemale->prevFemale=ptr->prevFemale;
        ptr=ptr->prevMale;
 }
```

```
    /*Delete less preferable nodes out of female's list*/
    ptr= &pair[maleProposer][female];
    ptr->nextMale=femaleHeader[female];
    femaleHeader[female]->prevMale=ptr;
    getch();

  printf("\n==========================================================
  =====");
    printf("\nRevised preference lists\n");

  printf("============================================================
  ====\n");
    printPreferenceLists();
  }
  }

  void rotations()
  {
  /* Gusfield & Irving*/
  int i,x,m,mprime;
  int stack[51],sp,instack[51];
  int rhoMale[51],rhoFemale[51],rhoSize;
 pairT *ptr;

 for (i=1;i<=n;i++)
    instack[i]=0;
 sp=(-1);
 x=1;
 while (x<=n)
 {
   if (sp==(-1))
```

```
{
        while (x<=n && maleHeader[x]->nextFemale->female==femaleOptimal[x])
                x++;
        if (x<=n)
        {
                sp++;
                stack[sp]=x;
                instack[x]=1;
        }
}
if (sp>(-1))
{
        m=stack[sp];
        for (m=femaleHeader[maleHeader[m]->nextFemale->nextFemale->female]
                ->prevMale->male;
        !instack[m];
        m=femaleHeader[maleHeader[m]->nextFemale->nextFemale->female]
                ->prevMale->male)
        {
                sp++;
                stack[sp]=m;
                instack[m]=1;
        }
        mprime=stack[sp];
        sp--;
        instack[mprime]=0;
        rhoSize=1;
        rhoMale[0]=mprime;
        rhoFemale[0]=maleHeader[mprime]->nextFemale->female;
        while (m!=mprime)
        {
```

```
                mprime=stack[sp];
                sp--;
                instack[mprime]=0;
                rhoMale[rhoSize]=mprime;
                rhoFemale[rhoSize]=maleHeader[mprime]->nextFemale->female;
                rhoSize++;
        }
        printf("\n\nHit A Key To See Rotation");getch();

printf("\n==========================================================
=====");
        printf("\n\nFound a rotation:\n");

printf("\n==========================================================
=====");
        for (i=rhoSize-1;i>=0;i--)
                printf("\n(%d,%d)\n",rhoMale[i],rhoFemale[i]);
        /*List deletions*/
        for (i=rhoSize-1;i>=0;i--)
                while (femaleHeader[rhoFemale[i]]->prevMale-
>male!=rhoMale[(i+1)%rhoSize])
                {       printf("Delete male=%d
female=%d\n",femaleHeader[rhoFemale[i]]->prevMale->male,rhoFemale[i]);
                        ptr=femaleHeader[rhoFemale[i]]->prevMale;
                        ptr->prevFemale->nextFemale=ptr->nextFemale;
                        ptr->nextFemale->prevFemale=ptr->prevFemale;
                        ptr->prevMale->nextMale=ptr->nextMale;
                        ptr->nextMale->prevMale=ptr->prevMale;
                }
        printf("\nHit A Key To See Revised Preference List");getch();
```

```
printf("\n==========================================================
=====\n");
        printf("\nRevised preference lists:\n");

printf("\n==========================================================
=====\n");
        printPreferenceLists();
        printf("\nHit A Key To See The Next Mathching");getch();

printf("\n==========================================================
=====\n");
        printf("Next matching:\n");

printf("\n==========================================================
=====\n");
        printMatching();
  }
}
}

void main()
{

int i;
clrscr();
saveInput();
readInputRolesReversed();
printPreferenceLists();
galeShapley();
/*Save/print female-optimal solution*/
```

```
 for (i=1;i<=n;i++)
   femaleOptimal[i]=femaleHeader[i]->prevMale->male;
 printf("\n=================================================================
 =====\n");
 printf("Female Optimal Solution\n");
 printf("\n=================================================================
 =====\n");
 for (i=1;i<=n;i++)
   printf("%d %d\n",i,femaleOptimal[i]);
 readInput();
 printf("\n=================================================================
 =====\n");
 printf("Correct role inputs\n");
 printf("\n=================================================================
 =====\n");
 printPreferenceLists();
 galeShapley();
 /* Save male-optimal solution */
 for (i=1;i<=n;i++)
   maleOptimal[i]=maleHeader[i]->nextFemale->female;
 printf("\n=================================================================
 =====\n");
 printf("Male optimal solution:\n");
 printf("\n=================================================================
 =====\n");
 printMatching();
 rotations();
 printf("\n=================================================================
 =====\n");
 printf("***Thank You for using the Program Have A Nice Time***\n");
```

```
printf("\n==========================================================
=====\n");
getch();
}
```

4.2 Java Source Code

4.2.1 Stable. Java

```
/**
 * FILE: Stable. java
* This file defines the main stable marriage applet.
 */
import java.util.*;
import java.applet.*;
import java.awt.*;
import java.net.*;
import javax.swing.*;
public class Stable extends Applet
{
 int iCouples;
 Vector vMen;
 Vector vWomen;
 boolean fNotAllEngaged;
 Enumeration e;
 int state;
 Button bStep, bRun, bReset;
 TextArea tarea;
 String input;
 public void init()
 {
```

```
String s_Clip;
if( null != (s_Clip = getParameter( "SOUND" ) ) ) {
 AudioClip ac = getAudioClip( getDocumentBase(), s_Clip );
 ac.loop();
}
 input = JOptionPane.showInputDialog("Enter No Of Couples (less than 25 : ");
iCouples = Integer.parseInt(input);
vWomen = new Vector();
vMen = new Vector();
tarea = new TextArea( 6, 60 );
tarea.setEditable( false );
for(int i = 1; i <= iCouples; i++ ) {
 vWomen.addElement(
    new Woman( getParam( "Woman" + i, "Woman " + i ), tarea ) );
}
for(int i = 1; i <= iCouples; i++ ) {
 vMen.addElement( new Man( getParam( "Man" + i, "Man " + i ), tarea ) );
}
reset();
initUI();
repaint();
}
public void initUI()
{
//setBackground(Color.whitw);
setForeground( Color.black );
GridBagLayout gbl = new GridBagLayout();
GridBagConstraints gbc = new GridBagConstraints();
setLayout( gbl );
// Labels for Columns of Buttons
Label l_Women = new Label( "Women", Label.CENTER );
```

```
Label l_Men = new Label( "Men", Label.CENTER );
Insets insetW = new Insets( 1, 15, 1, 30 );
Insets insetM = new Insets( 1, 30, 1, 15 );
gbc.gridwidth = GridBagConstraints.RELATIVE;
gbc.gridheight = 1;
gbc.insets = insetW;
gbc.anchor = GridBagConstraints.WEST;
gbl.setConstraints( l_Women, gbc );
add( l_Women );
gbc.gridwidth = GridBagConstraints.REMAINDER;
gbc.anchor = GridBagConstraints.EAST;
gbc.insets = insetM;
gbl.setConstraints( l_Men, gbc );
add( l_Men );
// Columns of Buttons
Enumeration enWomen = vWomen.elements();
Enumeration enMen = vMen.elements();
gbc.fill = GridBagConstraints.NONE;
for( int i = 0; i < iCouples; i++ ) {
 gbc.gridwidth = GridBagConstraints.RELATIVE;
 gbc.insets = insetW;
 gbc.anchor = GridBagConstraints.WEST;
 PersonButton pbW = new PersonButton( (Woman)enWomen.nextElement() );
 gbl.setConstraints( pbW, gbc );
 add( pbW );
 gbc.gridwidth = GridBagConstraints.REMAINDER;
 gbc.insets = insetM;
 gbc.anchor = GridBagConstraints.EAST;
 PersonButton pbM = new PersonButton( (Man)enMen.nextElement() );
 gbl.setConstraints( pbM, gbc );
 add( pbM );
```

```
    }
    Panel pButtons = new Panel();
    GridBagLayout gblB = new GridBagLayout();
    GridBagConstraints gbcB = new GridBagConstraints();
    pButtons.setLayout( gblB );

    bStep = new Button( "Step" );
    bRun = new Button( "Run" );
    bReset = new Button( "Reset" );
    gbcB.gridwidth = 1;
    gbcB.insets = new Insets( 1, 3, 1, 3 );
    gbcB.anchor = GridBagConstraints.CENTER;
    gblB.setConstraints( bStep, gbcB );
    pButtons.add( bStep );
    gbcB.gridwidth = 1;
    gblB.setConstraints( bRun, gbcB );
    pButtons.add( bRun );
    gbcB.gridwidth = GridBagConstraints.REMAINDER;
    gblB.setConstraints( bReset, gbcB );
    pButtons.add( bReset );
    gbc.anchor = GridBagConstraints.CENTER;
    gbc.insets = new Insets( 2, 6, 0, 6 );
    gbl.setConstraints( pButtons, gbc );
    add( pButtons );
    pButtons.show();
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbl.setConstraints( tarea, gbc );
    add( tarea );
  }
  public void reset()
  {
```

```
        Random rand = new Random();
        rand.setSeed( new Date().getTime() );
        Enumeration en = vMen.elements();
        while( en.hasMoreElements() ) {
          Person p = (Person) en.nextElement();
          p.reset();
          Vector v = (Vector) vWomen.clone();
          p.randomPrefs( v, rand );
        }
        en = vWomen.elements();
        while( en.hasMoreElements() ) {
          Person p = (Person) en.nextElement();
          p.reset();
          Vector v = (Vector) vMen.clone();
          p.randomPrefs( v, rand );
        }
        tarea.setText( "" );
        fNotAllEngaged = true;
        e = null;
        state = 1;
        repaint();
        /*  */
        en = vMen.elements();
        while( en.hasMoreElements() )
      {
          Person p = (Person) en.nextElement();
          p.reset();
          Vector v = (Vector) vWomen.clone();
          p.randomPrefs( v, rand );
        }
        en = vWomen.elements();
```

```
    while( en.hasMoreElements() ) {
      Person p = (Person) en.nextElement();
      p.reset();
      Vector v = (Vector) vMen.clone();
      p.randomPrefs( v, rand );
    }
    repaint();
  }
  public void printAllPrefs()
  {
    Enumeration en = vMen.elements();
    while( en.hasMoreElements() ) {
      Person p = (Person) en.nextElement();
      p.printPrefs();
    }
    en = vWomen.elements();
    while( en.hasMoreElements() ) {
      Person p = (Person) en.nextElement();
      p.printPrefs();
    }
  }
  public void step()
  {
    switch( state )
    {
    case 1:
        if( fNotAllEngaged ) {
          fNotAllEngaged = false;
          e = vMen.elements();
          state = 2;
        }
```

```
        else {
          state = 4;
          printCouples();
          bRun.disable();
          bStep.disable();
        }
        break;
    case 2:
        if( e.hasMoreElements() ) {
          Man m = (Man) e.nextElement();
          if( !m.getEngaged() ) {
            m.propose();
            fNotAllEngaged = true;
          }
          else {
            step();
          }
        }
        else {
          e = vWomen.elements();
          state = 3;
        }
        break;
    case 3:
        if( e.hasMoreElements() ) {
          Woman w = (Woman) e.nextElement();
          if( w.hasSuitors() ) {
            w.decide();
          }
          else {
            step();
```

```
            }
          }
          else {
            state = 1;
          }
          break;
      case 4:
          break;
      }
    repaint();
  }
  public void findMatches()
  {
    while( state != 4 ) {
      step();
    }
  }
  public void printCouples()
  {
    Enumeration en = vMen.elements();
    while( en.hasMoreElements() ) {
      Man m = (Man) en.nextElement();
      m.print( m.getName() + " is engaged to " +
               m.getFiancee().getName() );
    }
  }

  public void paint( Graphics g )
  {
    Enumeration en = vMen.elements();
    while( en.hasMoreElements() ) {
```

```
    Man m = (Man)en.nextElement();
   if( m.getEngaged() ) {
  g.setColor( Color.red );
Point pM = m.getButton().location();
Dimension dM = m.getButton().size();
Point pF = m.getFiancee().getButton().location();
        Dimension dF = m.getFiancee().getButton().size();

        g.drawLine( pM.x,
                    pM.y + (dM.height / 2 ),
                    pF.x + dF.width,
                    pF.y + (dF.height / 2 ) );
    }
    else if( m.getProposed() ) {
        g.setColor( Color.blue );

        Point pM = m.getButton().location();
        Dimension dM = m.getButton().size();

        Point pF = m.getPossible().getButton().location();
        Dimension dF = m.getPossible().getButton().size();

        g.drawLine( pM.x,
                    pM.y + (dM.height / 2 ),
                    pF.x + dF.width,
                    pF.y + (dF.height / 2 ) );
    }
  }
}

public boolean handleEvent( Event evt )
```

```
{
 switch( evt.id )
  {
  case Event.ACTION_EVENT:
       if( evt.target == bReset ) {
        bRun.disable();
        bStep.disable();
        reset();
        bRun.enable();
        bStep.enable();
        return( true );
       }
       else if( evt.target == bRun ) {
        bRun.disable();
        bStep.disable();
        while( 4 != state ) {
         step();
        }
        return( true );
       }
       else if( evt.target == bStep ) {

        step();

        return( true );
       }
  default:
       return( super.handleEvent( evt ) );
  }
}
```

```
    private String getParam( String s, String use )
    {
     if( null == (s = getParameter( s ) ) ) {
      return( use );
     }
    return( s );
   }

   private int getParam( String s, int i )
   {
    try
     {
        return Integer.parseInt( getParameter( s ) );
     }
   catch( NumberFormatException ex )
    {
        return i;
    }
 }

 public String getAppletInfo()
 {
  return( "This applet designed and destroyed by:\n"
        + "Anurag Pareek\n" + "pareek.anurag@gmail.com\n" );
}

public String[][] getParameterInfo()
{
 String[][] s = new String[5][3] ;
 s[0][0] = "SOUND";
 s[0][1] = "File";
```

```
        s[0][2] = "Au file to play in the background";
        s[1][0] = "COUPLES";
        s[1][1] = "Integer";
        s[1][2] = "A number of couples to create";
        s[2][0] = "MAN#";
        s[2][1] = "String";
        s[2][2] = "the name of man #";
        s[3][0] = "WOMAN#";
        s[3][1] = "String";
        s[3][2] = "the name of woman #";
        s[4][0] = "MESSAGEWIDTH";
        s[4][1] = "Integer";
        s[4][2] = "the width (in characters) of the text box which displays output";

        return( s );
    }
}
```

4.2.2 AcceptGuy.java

```
/**
 * FILE    : AcceptGuy.java
 *
 * This file has a small list of things for a woman to say when she accepts a
 * man's marraige proposal.
 */
import java.util.*;
public class AcceptGuy
{
 static Random rand = null;
 // Redefine this to add more elements to the switch below.
```

```
    private static final int SWITCHSIZE = 1;
    String s;
    public AcceptGuy( String woman, String man )
    {
     if( null == rand ) {
      rand = new Random();
     }

     switch( Math.abs( rand.nextInt() % SWITCHSIZE ) )
      {
      case 0:
          s = "\"Oh " + man + ",\" says " + woman + ", \" I will marry you!\"";
          break;
      case 1:
          s = woman + " smiles at " + man + ", \"I accept your proposal!\"";
          break;
          }
   }
   public String toString()
   {
    return s;
   }
}
```

4.2.3 BetterGuy.java

```
/**
 * FILE: BetterGuy.java
* This file has a small list of things for a woman to say when she decides to
 * ditch her old fiancee.
 */
import java.util.*;

public class BetterGuy
{
 static Random rand = null;
 // redefine to add more elements to the switch
 private static final int SWITCHSIZE = 1;
 String s;

 public BetterGuy( String woman, String newMan, String oldMan )
 {
  if( null == rand ) {
   rand = new Random();
  }
    switch( Math.abs( rand.nextInt() % SWITCHSIZE ) )
    {
    case 0:
        s = "\"Oh, " + newMan + ", I like you so much better than\nmy fiancee, " +
oldMan + "\", " + woman + " says.";
        break;
    case 1:
        s = woman + " sees her eyes and says, \"" + newMan + " you are\njust so"
         + " clever.\"";
        break;
```

```
        }
    }
     public String toString()
    {
     return s;
    }
}
```

4.2.4 Man.java

```
/**
 * FILE: Man.java
* This file defines a Man, an extension of the person class with the ability
 * to propose marraige.
 */
import java.io.* ;
import java.util.* ;
import java.awt.* ;

public class Man extends Person
{
 int      at;
 boolean   fProposed;
 Woman     wo_Possible;

 public Man( String sNomen, TextArea ta )
 {
  super( sNomen, ta );
  i_Sex = Person.MALE;
  at = 0;
  setProposed( false );
```

```
    setPossible( null );
   }

   public void reset()
   {
    super.reset();
    at = 0;
   }
    public void propose()
   {
    if( !getEngaged() ) {
     print( "Since " + getName() + " is not engaged, he proposes to the " + (at+1) + "
woman on his list." );
     setPossible( (Woman) vPreferences.elementAt( at ) );
     print( new Proposal( getPossible().getName(), getName() ).toString() );
     at++;
         getPossible().askToMarry( this  );
     setProposed( true );
    }
   }
    public void accept( Woman woman )
   {
    setFiancee( woman );
    setEngaged( true );
      print( getName() + " is now engaged to " +
           fiancee.getName() );
    setPossible( null );
    setProposed( false );
   }
   public void reject()
   {
```

```
    print( new Rejection( getPossible().getName(), getName() ).toString() );
    setPossible( null );
    setProposed( false );
   }
   public boolean getProposed()
   {
    return( fProposed );
   }
    public void setProposed( boolean b )
   {
    fProposed = b;
   }
    public Woman getPossible()
   {
    return( wo_Possible );
   }
  public void setPossible( Woman w )
  {
   wo_Possible = w;
  }
}
```

4.2.5 ManFrame.java

```
/**
 * FILE: ManFrame.java
 * This file defines a ManFrame, a frame with info about a man.
 */
import java.awt.*;
import java.awt.List;
import java.util.Enumeration;
```

```
public class ManFrame extends Frame
{
 Man m;
 Button bClose;
 public ManFrame( Man m )
 {
  this.m = m;
  setBackground( Color.white );
  setForeground( Color.black );
  GridBagLayout gbl = new GridBagLayout();
  GridBagConstraints gbc = new GridBagConstraints();

  setLayout( gbl );
  Font f = new Font( "Helvetica", Font.BOLD, 16 );
  Label name = new Label( m.getName(), Label.CENTER );
  name.setFont( f );
  gbc.gridwidth = GridBagConstraints.REMAINDER;
  gbl.setConstraints( name, gbc );
  add( name );
 Label spacer1 = new Label( "" );
 gbc.gridwidth = GridBagConstraints.REMAINDER;
 gbl.setConstraints( spacer1, gbc );
 add( spacer1 );
 if( m.getEngaged() ) {
  Label fiancee = new Label( "Fiancee: ", Label.CENTER );
  gbc.gridwidth = 1;
  gbl.setConstraints( fiancee, gbc );
  add( fiancee );
  PersonButton pb = new PersonButton( m.getFiancee(), true );
  gbc.gridwidth = GridBagConstraints.REMAINDER;
  gbl.setConstraints( pb, gbc );
```

```
    add( pb );
  }
  else if( m.getProposed() ) {
    Label possible = new Label( "Proposed To: ", Label.CENTER );
    gbc.gridwidth = 1;
    gbl.setConstraints( possible, gbc );
    add( possible );
    PersonButton pb = new PersonButton( m.getPossible(), true );
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    gbl.setConstraints( pb, gbc );
    add( pb );
  }
  else {
    Label none = new Label( "No fiancee or proposals.", Label.CENTER );
    gbl.setConstraints( none, gbc );
    add( none );
  }
  Label spacer2 = new Label( "" );
  gbc.gridwidth = GridBagConstraints.REMAINDER;
  gbl.setConstraints( spacer2, gbc );
  add( spacer2 );
  Label preferences = new Label( "Preferences:", Label.LEFT );
  gbl.setConstraints( preferences, gbc );
  add( preferences );
  List list = new List();
  Enumeration e = m.getPreferences();
  while( e.hasMoreElements() ) {
    list.addItem( ((Person)e.nextElement()).getName() );
  }
  gbc.weighty = 15;
  gbc.gridheight = 15;
```

```
    gbc.fill = GridBagConstraints.VERTICAL;
    gbl.setConstraints( list, gbc );
    add( list );
    Label spacer3 = new Label( "" );
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    gbc.weighty = 1;
    gbc.gridheight = 1;
    gbc.fill = GridBagConstraints.NONE;
    gbl.setConstraints( spacer3, gbc );
    add( spacer3 );

    bClose = new Button( "Close" );
    gbl.setConstraints( bClose, gbc );
    add( bClose );

    pack();
    resize( 200, 400 );
    show();
  }
  public boolean handleEvent( Event evt )
  {
    if( evt.id == Event.WINDOW_DESTROY ||
        ( evt.id == Event.ACTION_EVENT && evt.target == bClose ) ) {
      dispose();
      return( true );
    }
    return( super.handleEvent( evt ) );
  }
}
```

4.2.6 Person.java

```
/**
 *  FILE: Person.java
*  Program Based on Gale-Shapley Algorithm
* This file defines a Person, the super class of Man and Woman.
 */
import java.util.* ;
import java.awt.*;
public class Person
{
 protected Vector vPreferences;
 protected String sName;
 protected Person fiancee;
 protected boolean fEngaged;
 protected int   i_Sex;
 protected PersonButton pButton;
 protected TextArea tarea;
  public final static int MALE = -1;
 public final static int FEMALE = 1;
 public Person( String sName, TextArea ta )
 {
  this.sName = sName;
  tarea = ta;
  vPreferences = new Vector();
  setFiancee( null );
  setEngaged( false );
  setButton( null );
 }
public void reset()
{
```

```
    vPreferences = new Vector();
    setFiancee( null );
    setEngaged( false );
  }
    public void randomPrefs( Vector v, Random rand )
  {
    while( !v.isEmpty() ) {
      int i = Math.abs( rand.nextInt() ) % v.size();
      Object o = v.elementAt( i );
      v.removeElementAt( i );
      vPreferences.addElement( o );
    }
  }
  public void printPrefs()
  {
    print( sName + "\'s Preferences are:" );
   Enumeration e = vPreferences.elements();
    while( e.hasMoreElements() ) {
      Person p = (Person) e.nextElement();
          print( p.getName() );
    }
    print( "" );
  }
  public void dump()
  {
    print( getFiancee().getName() + " breaks off the enagagement with " +
            getName() );

    setFiancee( null );
    setEngaged( false );
  }
```

```
public void setFiancee( Person p )
{
 fiancee = p;
}
 public Person getFiancee()
{
 return( fiancee );
}
 public void setEngaged( boolean b )
{
 fEngaged = b;
}
 public boolean getEngaged()
{
 return( fEngaged );
}
public void setName( String s )
{
 sName = s;
}
 public String getName()
{
 return( sName );
}
public void setSex( int i )
{
 i_Sex = i;
}
public int getSex()
{
 return( i_Sex );
```

```
    }
    public PersonButton getButton()
    {
     return( pButton );
    }
    public void setButton( PersonButton pb )
    {
     pButton = pb;
    }
     public Enumeration getPreferences()
    {
     return( vPreferences.elements() );
    }
   void print( String s )
   {
    tarea.appendText( "\n" + s );
    int len = tarea.getText().length();
    tarea.select( len, len );
  }
}
```

4.2.7 PersonButton.java

```
/**
 *  FILE: PersonButton.java
*  This file defines a PersonButton, a button that when pressed will pop-up a  Frame for
the person. */

import java.awt.*;
public class PersonButton extends Button
{
```

```
    Person p;
    public PersonButton( Person p )
    {
     super( p.getName() );
     this.p = p;
     p.setButton( this );
    }
    public PersonButton( Person p, boolean bugFix )
    {
     super( p.getName() );
     this.p = p;
     }
   public boolean handleEvent( Event evt )
   {
    if( evt.id == Event.ACTION_EVENT ) {
     if( p.getSex() == Person.MALE ) {
          new ManFrame( (Man)p );
     }
     else
{
          new WomanFrame( (Woman)p);
     }
        return( true );
    }
   return( super.handleEvent( evt ) );
  }
}
```

4.2.8 Proposals.java

```
/**
 *  FILE: Proposals.java
*  This file defines a a set of things for a man to say to a woman when he
 *  proposes.
 */
import java.util.*;
public class Proposal
{
 static Random rand = null;
 // redefine to add more sayings.
 private static final int SWITCHSIZE = 1;
 String s;
 public Proposal( String woman, String man )
 {
  if( null == rand ) {
    rand = new Random();
  }
     switch( Math.abs( rand.nextInt() % SWITCHSIZE ) )
   {

   case 0:

        s = "\"" + woman + ",\" says " + man + ", \"Will you marry me?\"";
        break;

   case 1:

        s = "\"We belong together,\" " + man + " tells " + woman + "\n"
         + "\"Will you marry me?\"";
```

```
            break;
           }
    }
     public String toString()
    {
     return s;
    }
}
```

4.2.9 Rejection.java

```
 /**
 *  FILE: Rejection.java
* This file defines a a set of ways for a woman to reject a man
 */
import java.util.*;
public class Rejection
{
 static Random rand = null;
 // redefine to add more sayings
 private static final int SWITCHSIZE = 1;
 String s;
 public Rejection( String woman, String man )
 {
  if( null == rand ) {
    rand = new Random();
  }
     switch( Math.abs( rand.nextInt() % SWITCHSIZE ) )
    {
    case 0:
```

```
            s = "\"I'm sorry,\" " + woman + " tells " + man + ", \"It wouldn\'t work out for
us.\"";
            break;
        case 1:
            s = woman + " laughs at " + man + ", \"I would never accept your proposal!\"";
            break;
           }
  }
  public String toString()
  {
   return s;
  }
}
```

4.2.10 Woman.java

```
/**
 *  FILE: Woman.java
  *  This file defines a Woman, an extension of the person class with the ability
 *  to be proposed to, to accept and reject lists of proposals for marraige. */
import java.io.* ;
import java.util.* ;
import java.awt.* ;
public class Woman extends Person
{
 protected Vector vSuitors;
 public Woman( String sNomen, TextArea ta )
 {
  super( sNomen, ta );
  i_Sex = Person.FEMALE;
  vSuitors = new Vector();
```

```
}
public Enumeration getProposals()
{
 return( vSuitors.elements() );
}
public void askToMarry( Man man )
{
 vSuitors.addElement( man );
}

public void decide()
{
 Man man;
 if( 0 == vSuitors.size() ) {
    return;
 }
 if( !getEngaged() ) {
  man = (Man) vSuitors.elementAt( 0 );
  vSuitors.removeElementAt( 0 );
 }
 else {
  man = (Man) getFiancee();
 }
  Enumeration e = vSuitors.elements();
 while( e.hasMoreElements() ) {
  Man otherGuy =  (Man) e.nextElement();
  if( vPreferences.indexOf(otherGuy) < vPreferences.indexOf( man ) ) {
      if( man != (Man) getFiancee() ) {
       man.reject();
      }
      else {
```

```
            print( new BetterGuy( getName(), otherGuy.getName(),
    getFiancee().getName() ).toString() );
            getFiancee().dump();
            setFiancee( null );
            setEngaged( false );
          }
          man = otherGuy;
        }
        else {
          otherGuy.reject();
        }
      }
      if( man != (Man)getFiancee() ) {
        setFiancee( man );
        setEngaged( true );
        print( new AcceptGuy( getName(), getFiancee().getName() ).toString() );
        man.accept( this );
      }
      vSuitors = new Vector();
    }
    public boolean hasSuitors()
    {
      return( ( 0 < vSuitors.size() ) ? true : false );
    }
}
```

4.2.11 WomanFrame.java

```
/**
 * FILE: WomanFrame.java
* This file defines a a pop-up frame with info about a woman.
 */
import java.awt.*;
import java.awt.List;
import java.util.Enumeration;
public class WomanFrame extends Frame
{
 Woman w;
 Button bClose;
 public WomanFrame( Woman w )
 {
  this.w = w;
  setBackground( Color.white );
  setForeground( Color.black );
  GridBagLayout gbl = new GridBagLayout();
  GridBagConstraints gbc = new GridBagConstraints();
  setLayout( gbl );
  Font f = new Font( "Helvetica", Font.BOLD, 16 );
  Label name = new Label( w.getName(), Label.CENTER );
  name.setFont( f );
  gbc.gridwidth = GridBagConstraints.REMAINDER;
  gbl.setConstraints( name, gbc );
  add( name );
  Label spacer1 = new Label( "" );
  gbc.gridwidth = GridBagConstraints.REMAINDER;
  gbl.setConstraints( spacer1, gbc );
 add( spacer1 );
```

```
    if( w.getEngaged() ) {
      Label fiancee = new Label( "Fiancee: ", Label.CENTER );
      gbc.gridwidth = 1;
      gbl.setConstraints( fiancee, gbc );
      add( fiancee );
      PersonButton pb = new PersonButton( w.getFiancee(), true );
      gbc.gridwidth = GridBagConstraints.REMAINDER;
      gbl.setConstraints( pb, gbc );
      add( pb );
    }
    else {
      Label none = new Label( "No fiancee.", Label.CENTER );
      gbl.setConstraints( none, gbc );
      add( none );
    }
    Label spacer2 = new Label( "" );
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    gbl.setConstraints( spacer2, gbc );
    add( spacer2 );
    Label preferences = new Label( "Preferences:", Label.LEFT );
    gbl.setConstraints( preferences, gbc );
    add( preferences );
    List list = new List();
    Enumeration e = w.getPreferences();

    while( e.hasMoreElements() ) {
      list.addItem( ((Person)e.nextElement()).getName() );
    }
    gbc.weighty = 15;
    gbc.gridheight = 15;
    gbc.fill = GridBagConstraints.VERTICAL;
```

```
    gbl.setConstraints( list, gbc );
    add( list );
    Label spacer3 = new Label( "" );
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    gbc.fill = GridBagConstraints.NONE;
    gbl.setConstraints( spacer3, gbc );
    add( spacer3 );
    Label spacer4 = new Label( "" );
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    gbc.weighty = 1;
    gbc.gridheight = 1;
    gbc.fill = GridBagConstraints.NONE;
    gbl.setConstraints( spacer4, gbc );
    add( spacer4 );
    bClose = new Button( "Close" );
    gbl.setConstraints( bClose, gbc );
    add( bClose );
    pack();
    resize( 200, 450 );
    show();
  }
  public boolean handleEvent( Event evt )
  {
    if( evt.id == Event.WINDOW_DESTROY ||
          ( evt.id == Event.ACTION_EVENT && evt.target == bClose ) ) {
      dispose();
      return( true );
    }
    return( super.handleEvent( evt ) );
  }
}
```

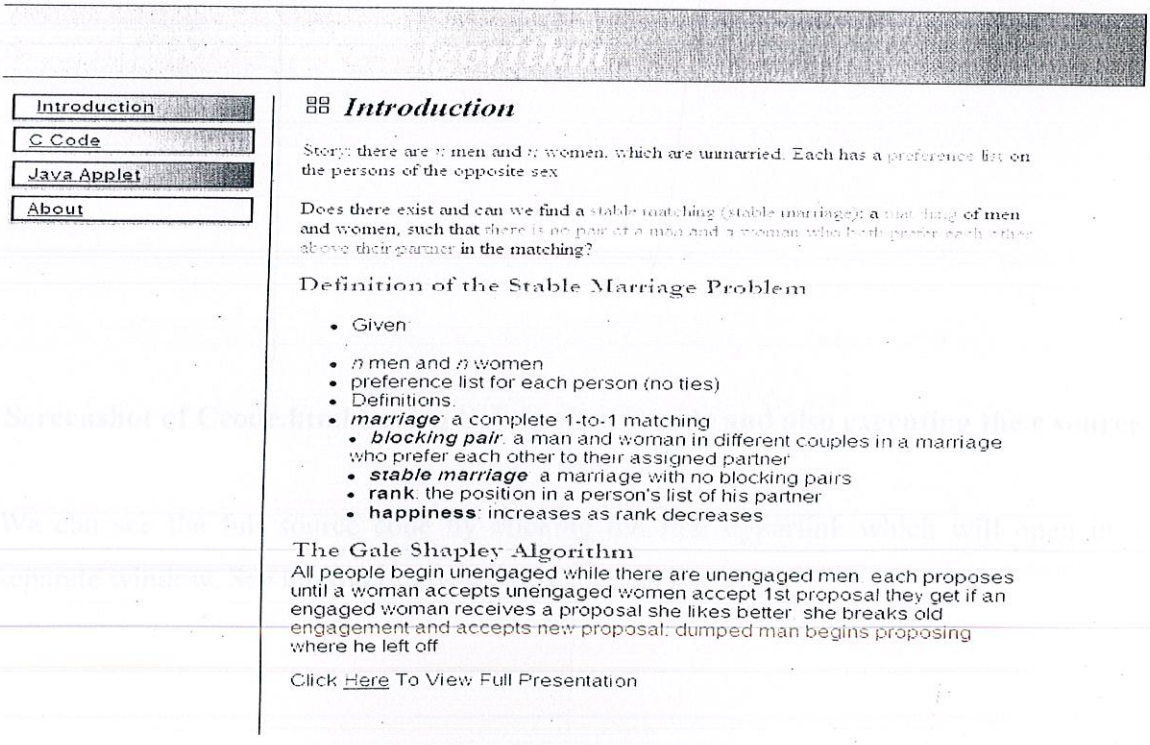
Chapter 5: Procedure and Layout

5.1 Executing Procedure of files

We had embedded the c source EXE file and the java applet on an offline website which can be stored on the local hard drive. It is essential to have TURBO C++ IDE installed on the system where you are going to test this code. On the contrary we rather don't want any JDK versions installed as we had already created the class files for each of the java files and had embedded that applet on our offline website.

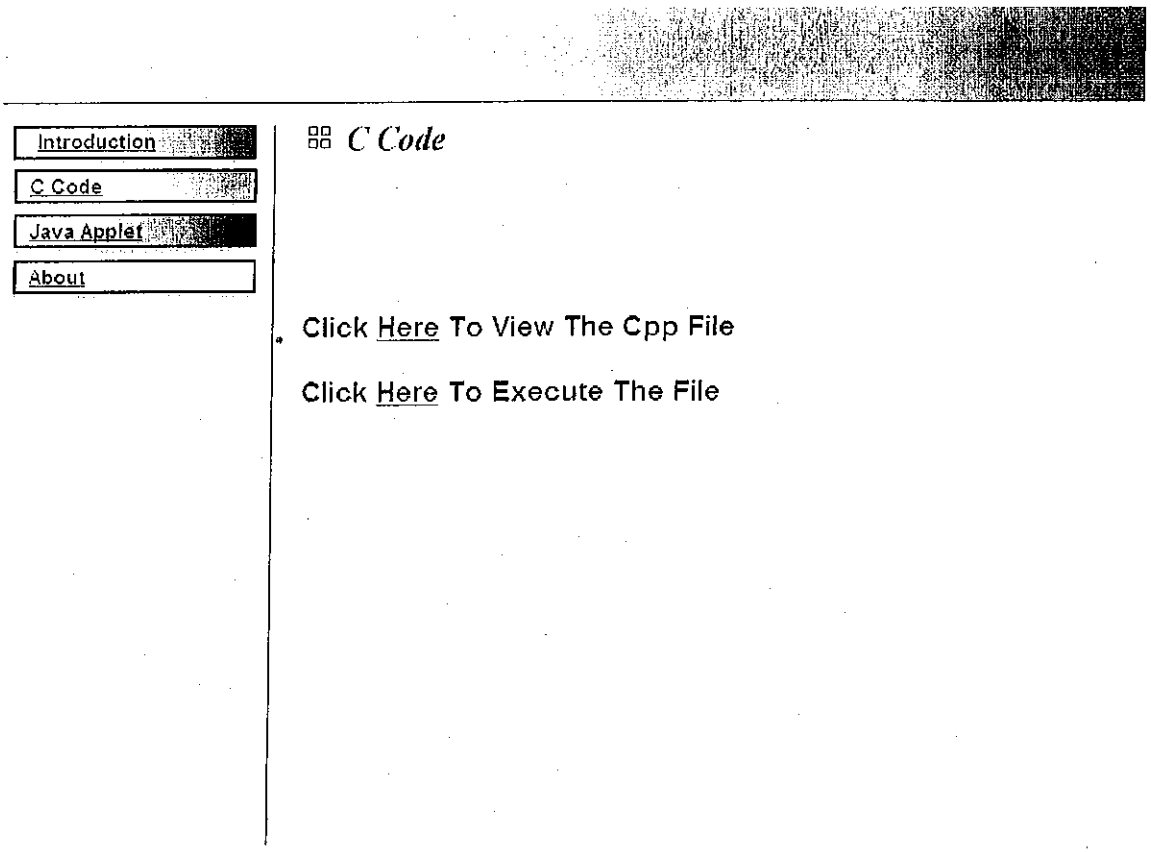
5.2 Layout of the offline website

The website comprises of four pages viz index.html, ccode.html, applet.html, about.html. When we open the index.html page we get a hyperlink for each of the above mentioned pages. The index.html contains a brief introduction of the stable marriage problem and a direct hyperlink to the full power point show of the problem.



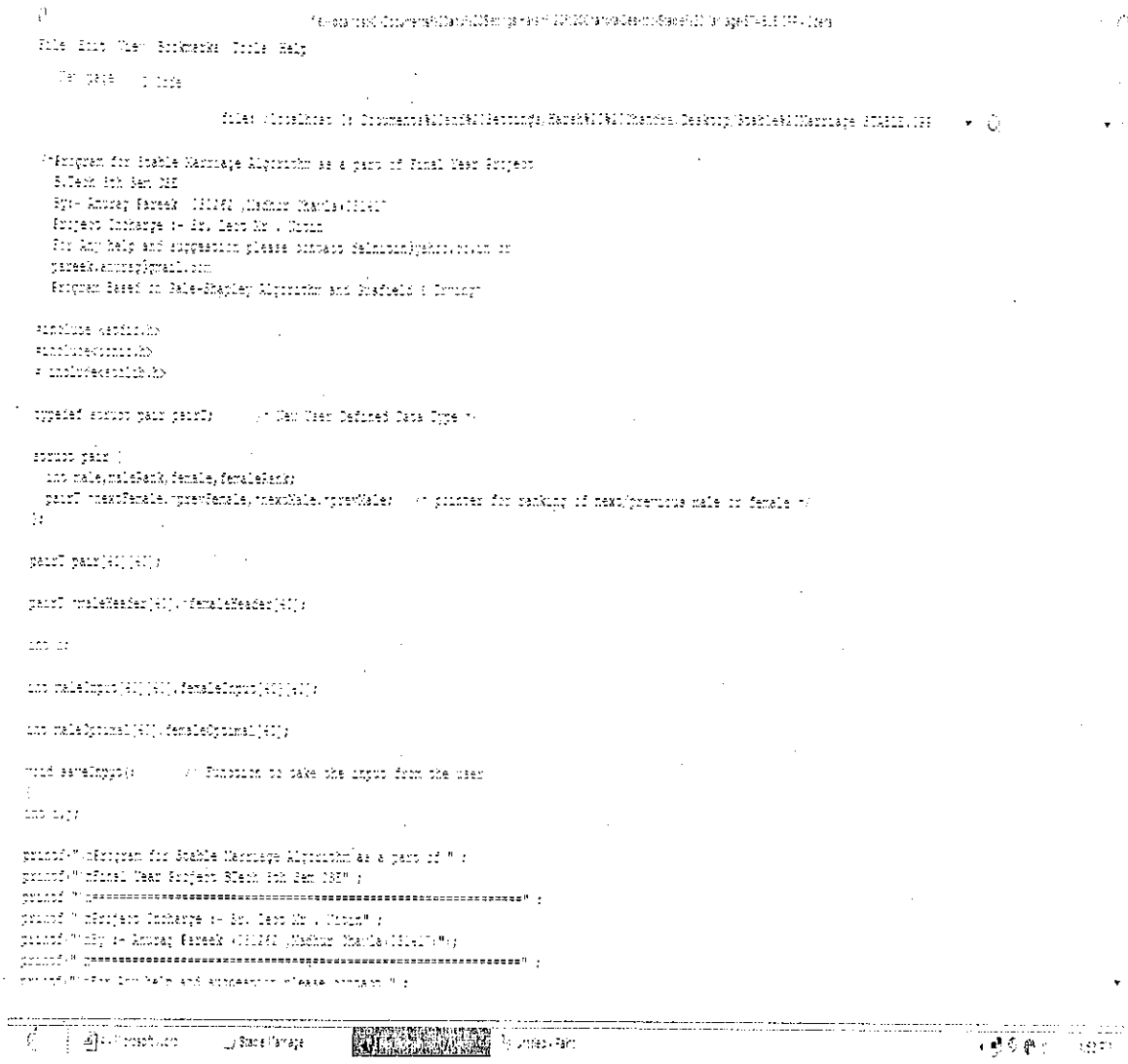
Screenshot of index, html showing brief introduction and link to full power point presentation

In figure we can see that on left hand side we have a link to other three pages. On clicking the C Code we can see the full source code of the stable marriage algorithm well written in c language. Also we can direct click to see the exe file and get the optimum matching by giving some input as needed.



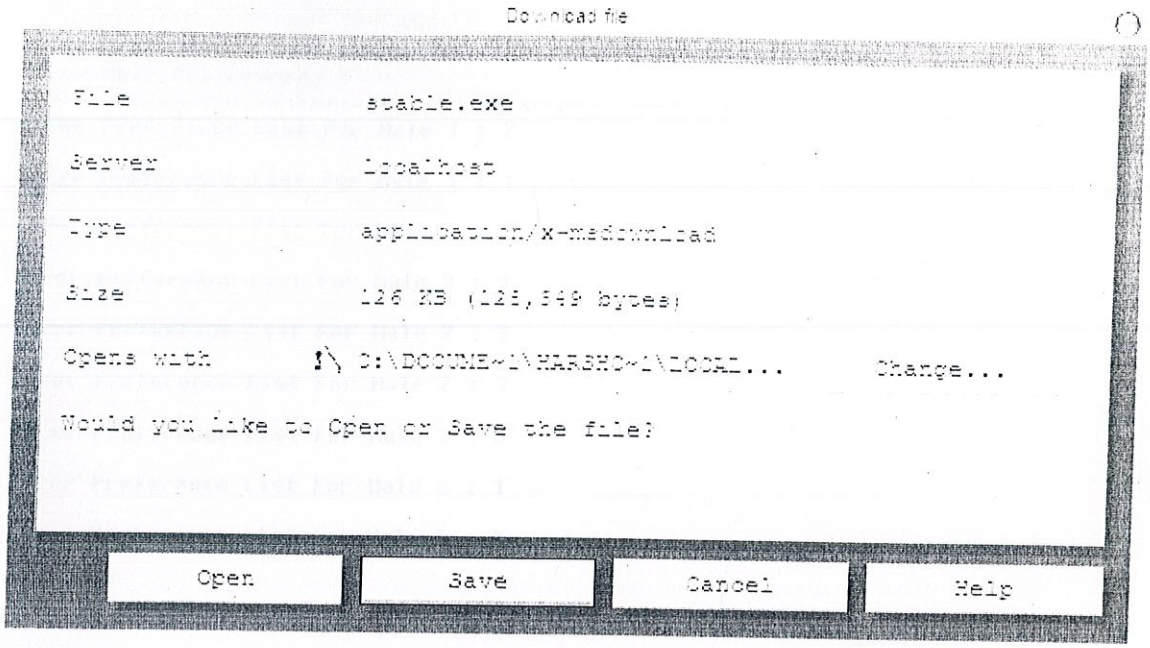
Screenshot of Ccode.html having link to view C code and also executing the c source file.

We can see the full source code by clicking the first hyperlink which will open in a separate window. See the figure given below.



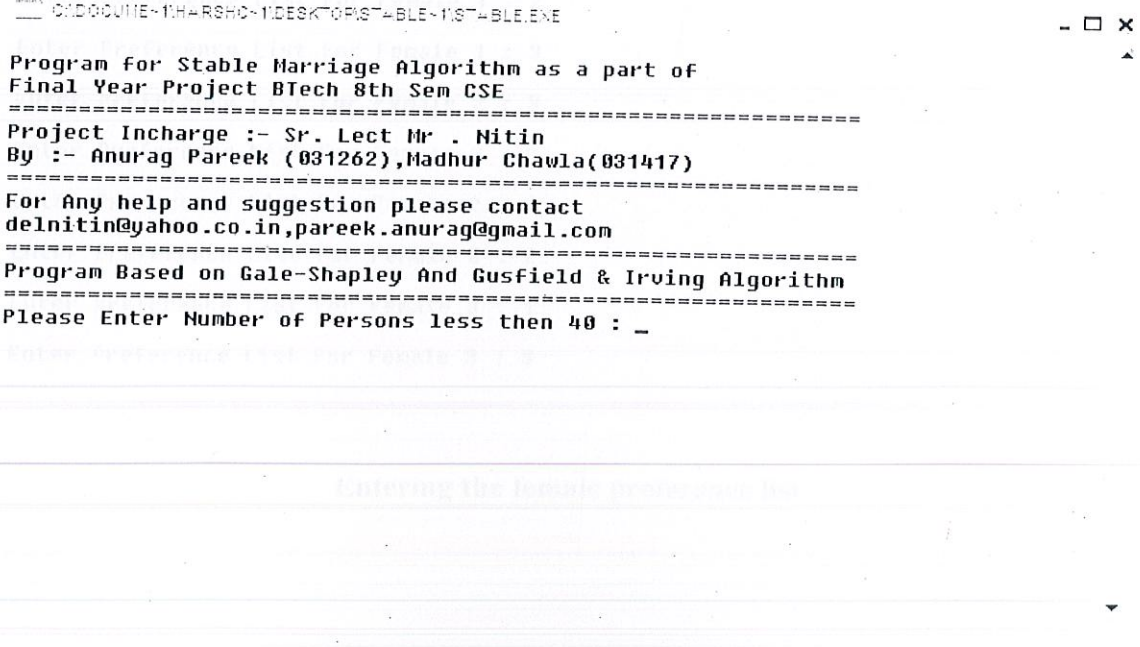
Screenshot showing the C source code opened in a separate window

When we click the EXE hyperlink a dialog box appears which ask us either to save or open the exe file. We do not need to save the file as it is already stored in the hard drive. On clicking the open button we can see the program in running. We have to make sure that the TURBO C++ IDE must be installed in our system. Lacking of which will lead to an unexpected error in running this EXE file.



Screenshot of the dialog box asking to save/open the EXE file

On opening the exe file we can run the program for any arbitrary input. For instance we are taking no of couples as 3 and will show the optimum matching according to the preference list given.



C Program in running asking for no of persons

```
C:\DOCUME~1\HARSHO~1\DESKTOP\STABLE~1\STABLE.EXE
================================================================
Enter Male Preferences :
================================================================
Enter Preference List For Male 1 : 2
Enter Preference List For Male 1 : 1
Enter Preference List For Male 1 : 3
Enter Preference List For Male 2 : 1
Enter Preference List For Male 2 : 2
Enter Preference List For Male 2 : 3
Enter Preference List For Male 3 : 3
Enter Preference List For Male 3 : 1
Enter Preference List For Male 3 : 2
```

Entering the male preference list

```
C:\DOCUME~1\HARSHO~1\DESKTOP\STABLE~1\STABLE.EXE
================================================================
Enter Female Preferences :
================================================================
Enter Preference List For Female 1 : 1
Enter Preference List For Female 1 : 2
Enter Preference List For Female 1 : 3
Enter Preference List For Female 2 : 3
Enter Preference List For Female 2 : 2
Enter Preference List For Female 2 : 1
Enter Preference List For Female 3 : 2
Enter Preference List For Female 3 : 1
Enter Preference List For Female 3 : 3
```

Entering the female preference list

```
C:\DOCUME~1\HARSHC~1\DESKTOP\STABLE~1\STABLE.EXE
3: 3 1 2
Male 3 proposes to female 2

First engagement for female 2

================================================================
Revised preference lists
================================================================

================================================================
Male preference lists is :
================================================================
1: 1 2 3
2: 3 2 1
3: 2 1 3

================================================================

Female preference lists is :
================================================================
1: 2 1 3
2: 1 2 3
3: 3 1 2
Male 2 proposes to female 3
```

Proposals and revision in preference list

Similarly, proposals and revision is done first according to Gale-Shapley and then Irving theorem.

```
C:\DOCUME~1\HARSHC~1\DESKTOP\STABLE~1\STABLE.EXE
First engagement for female 1

Delete male=3 female=1

================================================================
Revised preference lists
================================================================

================================================================
Male preference lists is :
================================================================
1: 1 2 3
2: 3 2 1
3: 2 3

================================================================

Female preference lists is :
================================================================
1: 2 1
2: 1 2 3
3: 3 1 2

================================================================
```

Rotation for female optimal solution

```
C:\DOCUME~1\HARSHC~1\DESKTOP\STABLE~1\STABLE.EXE
==========================================================
Female Optimal Solution
==========================================================
1 1
2 3
3 2
==========================================================
Correct role inputs
==========================================================
==========================================================
Male preference lists is :
==========================================================
1: 2 1 3
2: 1 2 3
3: 3 1 2
==========================================================
Female preference lists is :
==========================================================
```

Female optimal solution

Now, the program will similarly do the rotation for male optimal and yield the final optimal matching.

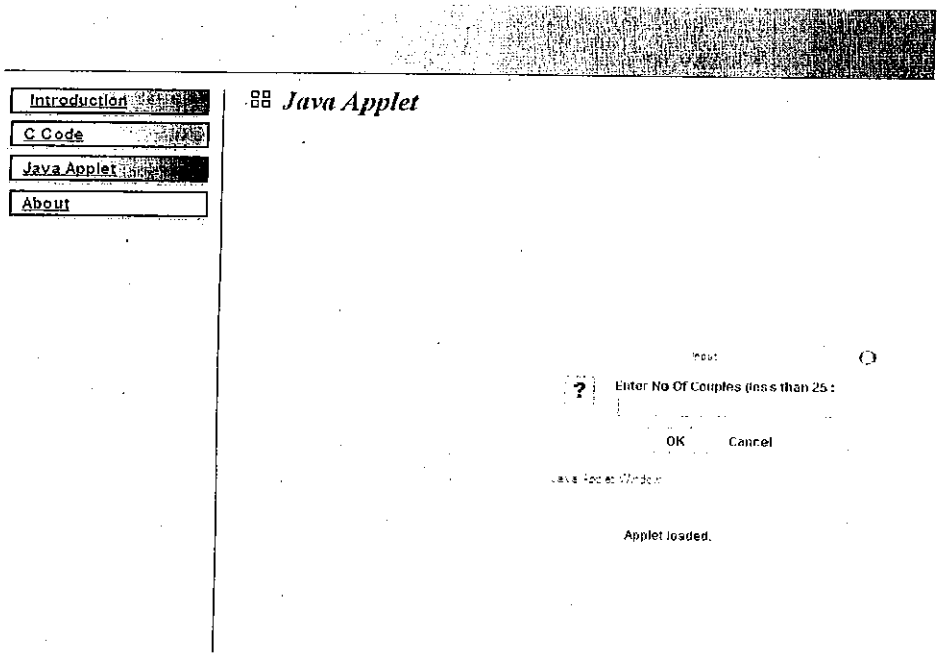
```
C:\DOCUME~1\HARSHC~1\DESKTOP\STABLE~1\STABLE.EXE
==========================================================
Male optimal solution:
==========================================================
1 2
2 1
3 3

Hit A Key To See Rotation
==========================================================
Found a rotation:
==========================================================
(1,2)

(2,1)
Delete male=1 female=2
Delete male=2 female=1

Hit A Key To See Revised Preference List
==========================================================
```

Male Optimal solution

```
2: 3
3: 2

=================================================================

Female preference lists is :
=================================================================
1: 1
2: 3
3: 2

Hit A Key To See The Next Mathching
=================================================================
Next matching:

=================================================================
1 1
2 3
3 2

=================================================================
***Thank You for using the Program Have A Nice Time***

=================================================================
```

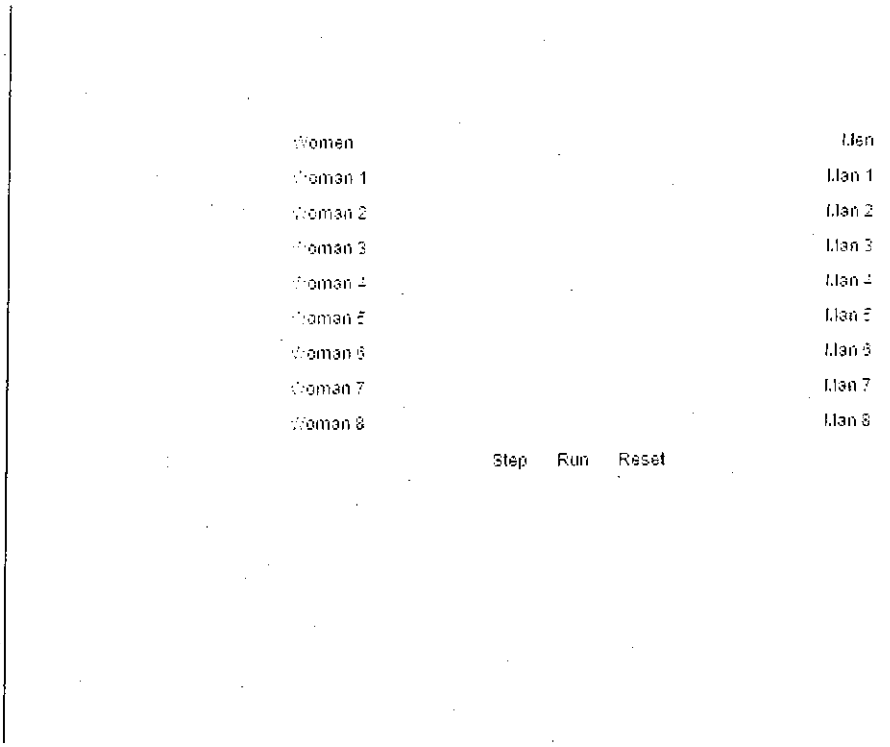
Final matching

The third page applet.html contains the java applet for the implementing stable marriage Gale-Shapley algorithm. It initially asks for the no of couples and then run the algorithm See the figure given below.



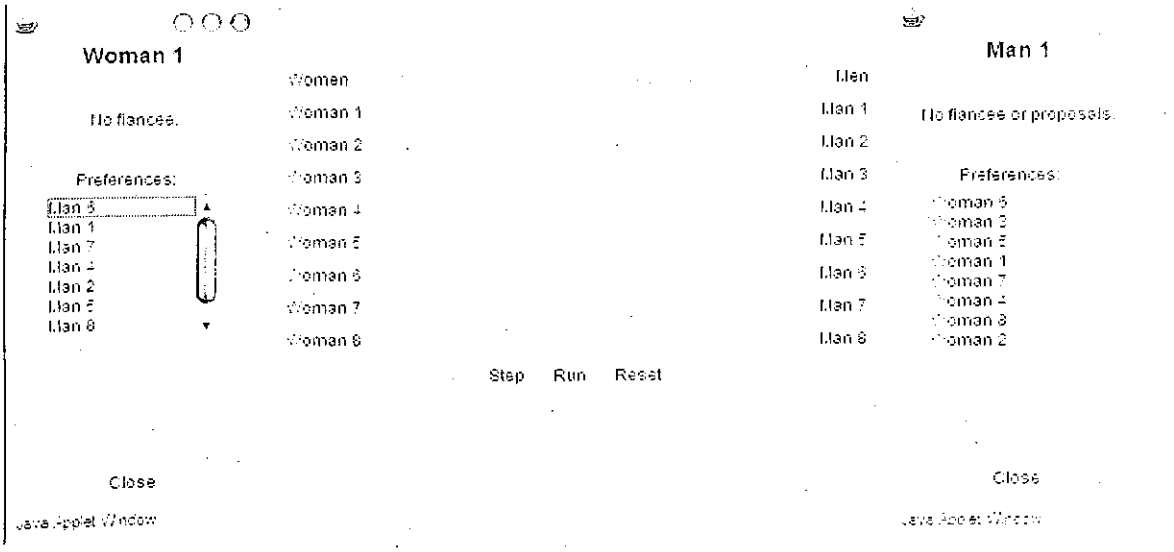
Applet asking for no of couples

After entering the no of couples (Say 8),the applet randomly assigns the preference list for each man and woman.



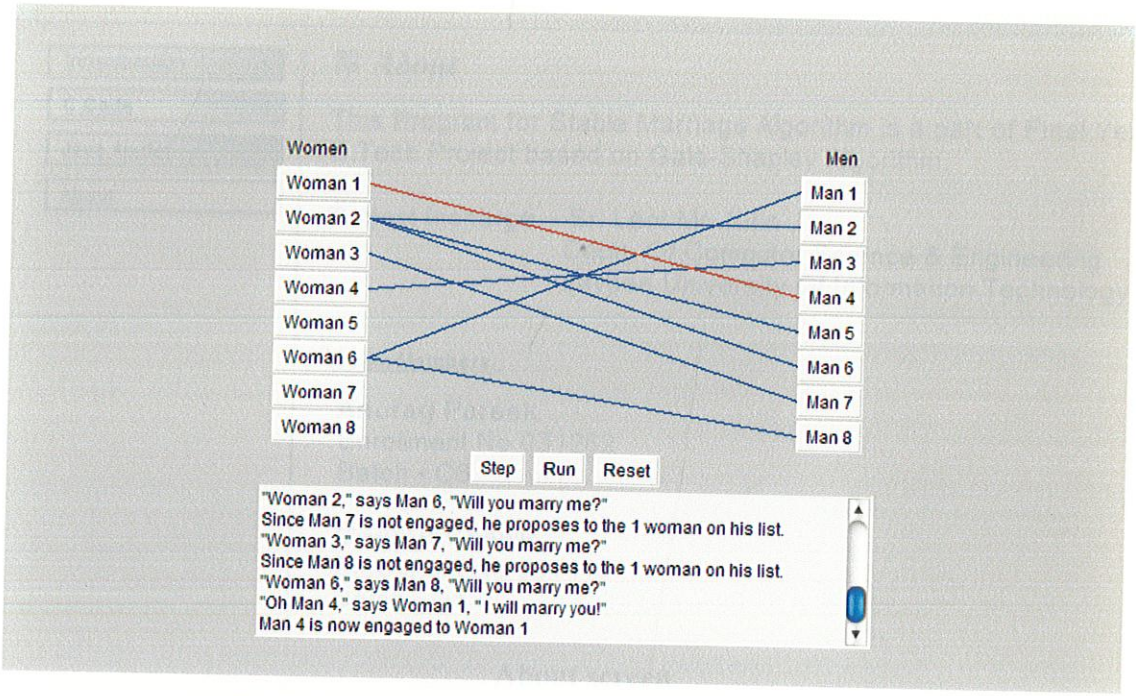
Applet Loaded

We can see the preference list of each man or woman by clicking the respective buttons.

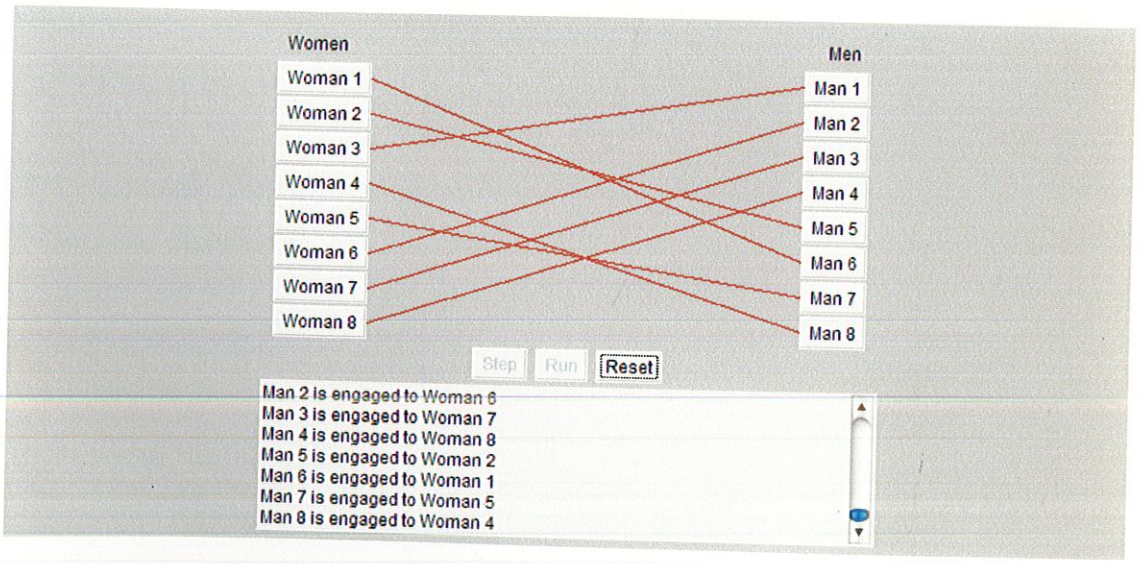


Preference list of first man and first woman

Now, we have two ways to run the applet. Either step by step or run the applet at once. However, in both cases the applet will show what it has done in the text area. The red line shows the final matching between the two couples and blue line shows that this is the initial matching not the optimal. See the figure below.

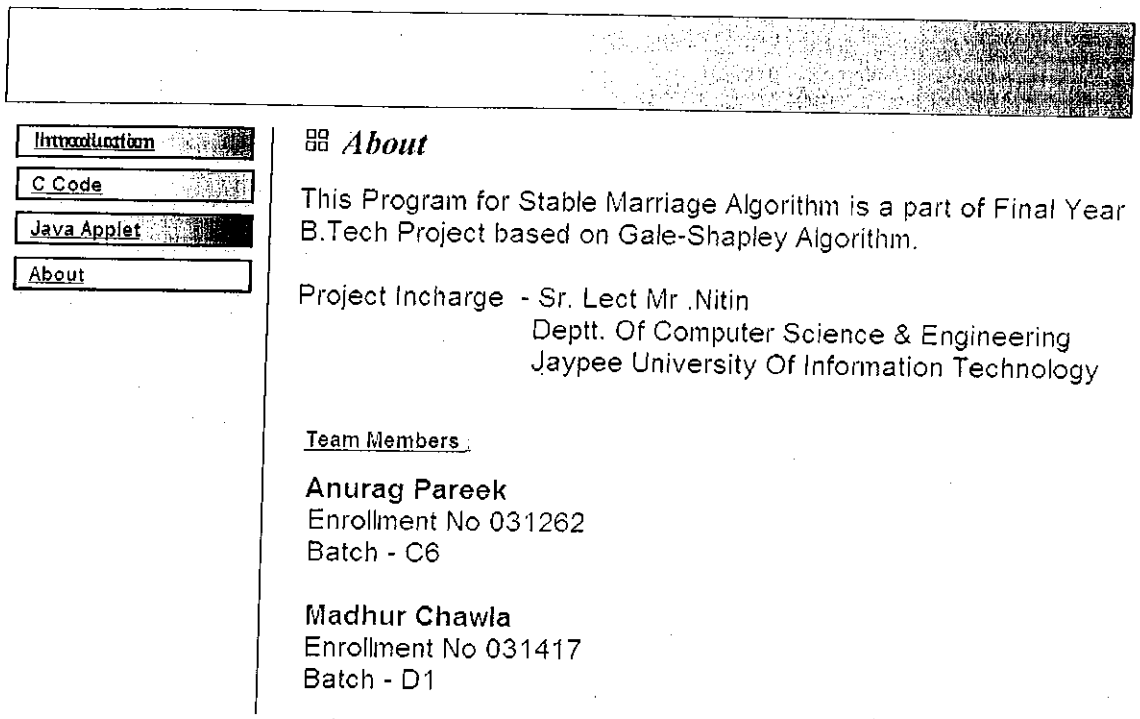


Initial and optimal matching of man and woman



Final matching of couples

Finally there is one more file called about.html which contains the details of the Team members and project coordinator. See the screenshot given below.



About screen

Bibliography

[1] Alan W. Biermann. An overview course in academic computer science: A new approach for teaching non majors. In The Papers of the Twenty-first SIGCSE Technical Symposium on Computer Science Education.

[2] Dan Gusfield and Robert W. Irving. The Stable Marriage Problem: Structures and Algorithms.

[3] Subramanian, A., and Nitin, "On a performance of multi-stage interconnection network," IEEE ADCOM, pp. 73-79, December 2004.

[4] Nitin, "On Analytic Bounds of Regular and Irregular Fault tolerant Multi-stage Interconnection Networks"

[5] Michael R. Garey and David S. Johnson. Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman andCompany, 1979.

[6] D. Gale and L.S. Shapley. College admissions and the stability of marriage. American Mathematical Monthly,

[7] D.E. Knuth. The Art of Computer Programming, volume 3. Addison-Wesley, 1973. Sorting and Searching.

[8] D.E. Knuth. Mariages Stables. Les Presses del'Universit_e Montr_eal, 1976.

[9] National Resident Matching Program,Evanston, Illinois. Handbook for students Participating Through U.S. Medical Schools, April1991.