Akkad Bakad Bambai Bo

The Josephus Problem

Shivam Sharma, Rajat Saini and Natasha Sharma
Cluster Innovation Center, University of Delhi

Abstract

We aim to give explanation of the recursive formula for the Josephus problem (also known as a popular game in India," Akkad Bakad Bambai Bo") when the elimination doesn't occur at constant k step but it is changed in a recursive manner. An application of the mentioned algorithm has been explained to select the cards of a particular suit by generating a sequence of steps with the algorithm to arrange the cards.

Introduction

The afore mentioned problem is named after Flavius Josephus, who with other 40 members was blocked by Romans in a cave and they all decided to suicide by forming a circle and killing themselves in step of three over capture by the Romans. Josephus states that by luck or possibly by the hand of God, he and another man remained the last and gave up to the Romans.

The Josephus problem is that a group of n people arranged in a circular format are numbered 1to n in succeeding manner. With step of 2 we start eliminating the people. For example, if n = 6, the people are removed in the order 2, 4, 6, 3, 1, and the last person remaining is no. 5. Let T(n) denote the last person remaining. The problem aims to find some simple way to compute T(n) for any positive integer $n \geq 1$ and step $k>0$.

Algorithm

k - represents the eliminating step
n - represents total number of people
T(i) –represents the survival's position when *i* is the total number of people
Recursive formula for constant step k=2 and n people:

The position of the survival in *T(2n)* when every second person out of n persons standing in a circle is eliminated until only one survives is given by the recurrence relation
$T(2n) = 2T(n)-1$ with $T(1) = 1$

Explanation: Let *i* be the position of any survivor element after first elimination. Clearly, the corresponding position in the circle having 2*n* elements is 2*i-1*. Hence *T(2n)=2T(n)-1*.

n	*survival position in n people is i*	*i*	*2n*	*Survival's position in 2n is 2i-1*
1	1	1	2	1
2	1 2	1	4	1
3	1 2 3	3	6	5
4	1 2 3 4	1	8	1
5	1 2 3 4 5	3	10	5
6	1 2 3 4 5 6	5	12	9
7	1 2 3 4 5 6 7	7	14	13

Table 1: shows the survival's position in T(2n)

The position of the survival in *T(2n+1)* when every second person out of n persons standing in a circle is eliminated until only one survives is given by the recurrence relation
$T(2n+1) = 2T(n)+1$ with $T(1) = 1$

Explanation: Let *i* be the position of any survivor element after first elimination. Clearly, the corresponding position in the first circle having 2n+1 elements is 2i+1. Hence T(2n+1)=2T(n)+1.

n	*survival position in n is i*	*i*	*2n+1*	*Survival's position 2i+1*
1	1	1	3	3
2	1 2	1	5	3
3	1 2 3	3	7	7
4	1 2 3 4	1	9	3
5	1 2 3 4 5	3	11	7
6	1 2 3 4 5 6	5	13	11
7	1 2 3 4 5 6 7	7	15	15

Table 2: shows survival's position for T(2n+1)

Another way of finding the survivor's position T(n) for n persons with constant step k=2 is binary method [6] :

Therefore, *T(n)*= left rotation of the binary digits of n
Suppose, for n (base 10) $= x_1 x_2 x_3 \ldots x_n$ (base 2) where x_k is the binary values of n with $x_1 = 1$.
Then,

$$T(n) = x_2 x_3 \ldots x_n x_1$$

For example , for n = 6 with k=2
$(6)_{10} = (110)_2$ since x_3 in binary expansion of $(6)_{10}$ is 1
So, the survival's position is $T(6) = (101)_2 = (5)_{10}$

Similar kinds of recursive formulas exits for *step (k) =r* $r \in N$

So, another way of computing the survivor's position *T(n)* is by using the dynamic recursive method[5] [7] which takes O(n) running time complexity.

Before using the formula let us fix some notations mentioned in [5] for the recursive formula.

Number the n positions in the circle by *1, 2,3… n* and start counting at number 1. Then every k^{th} element is removed. It is then defined as

$$T(n,k,i), \qquad (n>1;\ k>1;\ 1<i<n)$$

to be the number of the i^{th} element which is removed by the process described above (see the example in Figure a).

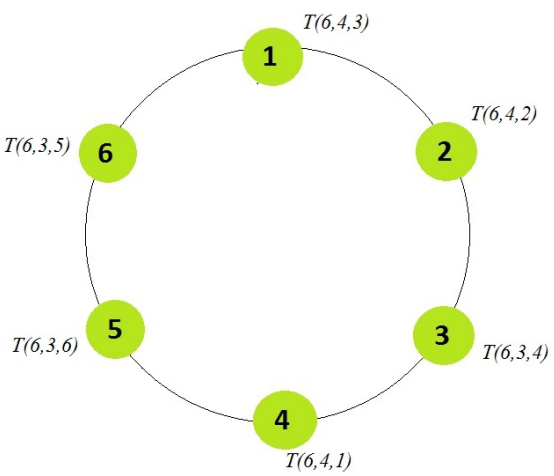

Fig a.

Here the *T(n,k,i)* denotes the person eliminated at i^{th} step .

In [5] the recursive formula for the *T(n, k, i)* has been explained and it states that for Josephus problem the following recursion holds:

$$T(n,k,i) = (\ T(n-1,k,i+1) + k-1\ ) \bmod (n) + 1\ , \qquad (n>1;\ k>1;\ 1<i<n)$$

With initial value T(1,k,i)=1 or T(n ,k ,1)=(k-1)mod(n) +1

Proved in [5]

"Suppose, we know the value of T(n, k, i) = g. Hence, if we start counting at number 1, the i^{th} member removed is number g. Now consider T(n, k, i +1). Because of T(n, k, 1) = (k - 1)mod(n), in the first step number (k-1)mod(n)+1 is removed and for the second step we start counting at number k+1. Therefore the problem is to find the i^{th} member which is removed (after removing (k - 1) mod (n)+1) when we start counting at number k+1. But this is (g + k-1) mod (n). So, we get the recursive formula."

If we know that *T(n,k,i) = g* ,then we start counting at 1 then the i^{th} member removed is g. Now we need to find the number removed in the next step when we start counting from k+1 number i.e. we need to find *T(m,k,i+1)=??*

So, the *T(m,k,i+1)=(g+k-1)%n* +1

For example, the survival position for *n=11* with constant step *k=3* is calculated and the output calculated recursively is shown in the *table 3*.

No of People left (n)	Element Remained	Survival's Position *f(n,k)=(f(n-1,k)+k-1)%n +1* *f(1,k)=1with initial k=3*
11	**1 2 3 4 5 6 7 8 9 10 11**	**f(11,3) = (f(10,3) + 2)%11+1 = 7**
10	**1 2 3 4 5 6 7 8 9 10**	f(10,3) = (f(9,3) + 2)%10 +1 = **4**
9	**1 2 3 4 5 6 7 8 9**	f(9,3) = (f(8,3) + 2)%9 +1 = **1**
8	**1 2 3 4 5 6 7 8**	f(8,3) = (f(7,3) + 2)%8 +1 = **7**
7	**1 2 3 4 5 6 7**	f(7,3) = (f(6,3) + 2)%7 +1 = **4**
6	**1 2 3 4 5 6**	f(6,3) = (f(5,3) + 2)%6 +1 = **1**
5	**1 2 3 4 5**	f(5,3) = (f(4,3) + 2)%5 +1 = **4**
4	**1 2 3 4**	f(4,3) = (f(3,3) + 2)%4 +1 = **1**
3	**1 2 3**	f(3,3) = (f(2,3) + 2)%3 +1 = **2**
2	**1 2**	f(2,3) = (f(1,3) + 2)%2 +1 = **2**
1	**1**	f(1,3) = **1**

Table 3: shows survival's position for constant k=3 elimination step using the dynamic programming method

Mathematical Analysis

Extended Josephus Problem

Another Josephus problem is to determine the survivor's position *T(n)* with variable steps *k* . Survivor's position can easily be calculated with slight modification in the dynamic recursive method.

Steps changing as counting:

For n people with steps changing as a counting sequence
For instance k takes the sequences 2, 3, 4, 5, 6… after elimination of person at each step with initial step k=2
So, the dynamic recursive formula becomes

$$f(n,k) = (f(n-1,k+1,i+1) + k -1)\%n +1$$

For example, the survival's position for *n=11* people with initial step *k=3* is calculated using the dynamic programming method and the output calculated iteratively is shown in the *table 5*.

No of People left (n)	Element Remained	Survival's Position *f(n,k)=(f(n-1,k+1,i+1)+k-1)%n +1*

		f(1,k)=1with initial k=3 , i=1
11	**1 2 3 4 5 6 7 8 9 10 11**	**f(11,3,1) = (f(10,4,2) + 2)%11 +1=10**
10	**1 2 3 4 5 6 7 8 9 10**	f(10,4,2) =(f(10,4,2) + 2)%11 +1= **7**
9	**1 2 3 4 5 6 7 8 9**	f(9,5,3) =(f(10,4,2) + 2)%11 +1= **3**
8	**1 2 3 4 5 6 7 8**	f(8,6,4) =(f(10,4,2) + 2)%11 +1= **7**
7	**1 2 3 4 5 6 7**	f(7,7,5) = (f(10,4,2) + 2)%11 +1=**1**
6	**1 2 3 4 5 6**	f(6,8,6) = (f(10,4,2) + 2)%11 +1=**1**
5	**1 2 3 4 5**	f(5,9,7) = (f(10,4,2) + 2)%11 +1=**5**
4	**1 2 3 4**	f(4,10,8) =(f(10,4,2) + 2)%11 +1= **1**
3	**1 2 3**	f(3,11,9) = (f(10,4,2) + 2)%11 +1=**3**
2	**1 2**	f(2,12,10) =(f(10,4,2) + 2)%11 +1= **1**
1	**1**	**1**

Table 4: shows survival's position with steps as sequence and initial k=3 elimination step using the dynamic programming method

Steps changing as multiples:
For instance k takes the values with initial step k=2 as 2, 4, 6, 8…
With initial step k=3, takes the values 3, 6, 9…
So, the dynamic recursive formula becomes

f(n,k) = (f(n-1,k,i+1) + k*i-1)%n +1 where 0< i <n

For example, the survival's position for *n=11* people with initial step *k=3* is calculated using the dynamic programming method and the output calculated iteratively is shown in the *table 5.*

No of People left (n)	Element Remained	Survival's Position *f(n,k)=(f(n-1,k,i+1)+k*i-1)%n +1* *f(1,k)=1with initial k=3*
11	**1 2 3 4 5 6 7 8 9 10 11**	**f(11,3,1) = (f(10,3,2) + 2) %11+1 = 6**
10	**1 2 3 4 5 6 7 8 9 10**	f(10,3,2) = (f(9,3,3) + 5)%10 +1 **= 3**
9	**1 2 3 4 5 6 7 8 9**	f(9,3,3) = (f(8,3,4) + 8)%9 +1 **= 7**
8	**1 2 3 4 5 6 7 8**	f(8,3,4) = (f(7,3,5) + 11)%8 +1 **= 7**
7	**1 2 3 4 5 6 7**	f(7,3,5) = (f(6,3,6) + 14)%7 +1 **= 3**
6	**1 2 3 4 5 6**	f(6,3,6) = (f(5,3,7) + 17)%6 +1 = **2**
5	**1 2 3 4 5**	f(5,3,7) = (f(4,3,8) + 20)%5 +1 = **2**
4	**1 2 3 4**	f(4,3,8) = (f(3,3,9) + 23)%4 +1 = **1**
3	**1 2 3**	f(3,3,9) = (f(2,3,10) + 26)%3 +1 = **1**
2	**1 2**	f(2,3,10) = (f(1,3,11) + 29)%2 +1 = **1**
1	**1**	1

(Table 5: shows survival's position with multiples of the initial k=3 elimination step using the dynamic programming method)

Implementation of the Algorithm

A game based on the Josephus Algorithm has been built to predict the survival's position and a GUI to depict the Josephus game simulation

Fig 1.GUI layout of the game

The game has two options either a person can play the simulation or he can predict the survival position and the graphical user interface would give him the correct answer with different options of eliminating steps.

Playing the simulation for the Josephus Problem

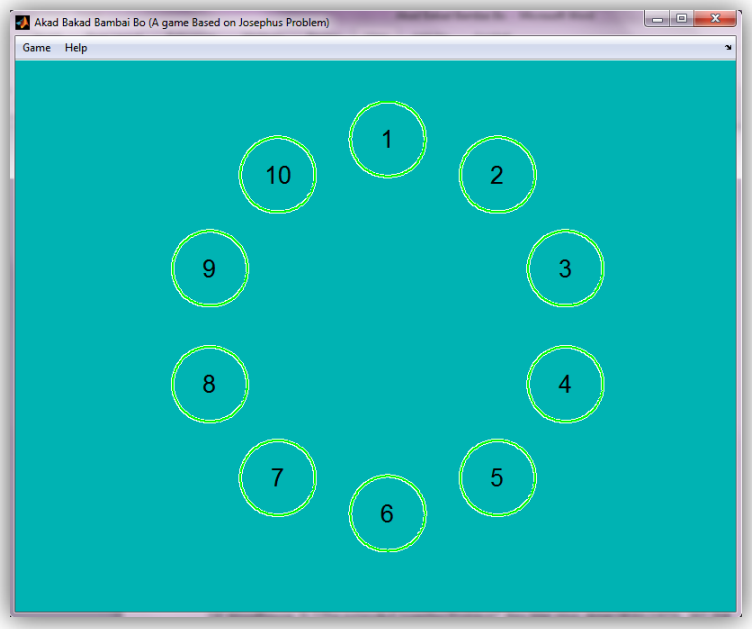

Fig 2.The simulation layout

The outputs after elimination, eliminated positions are marked with red circles and the green circle shows the survived person in the figure 3.

Fig 3.figure after elimination of all persons and with only one survived person and with the output table on the right side depicting the person eliminated at each step

Problem 2: Select the cards of a particular suit using Josephus Problem10
Solution:
We can generate the sequence of selection using the code given in appendix code 1:
So according to the sequence generated we can arrange the cards in the order of selection as shown in the table below.

Sequence Generated	2	4	6	8	10	12	1	5	9	13	7	3	11
Cards Arrangement	Ace	King	Queen	Joker	5^{th} card	6^{th} card	7^{th} card	8^{th} card	9^{th} card	10^{th} card	11^{th} card	12^{th} card	13^{th} card

Where n=13 and k=2(constant) , also we can generate sequence with sequence as sum and sequence as multiple using the same code.

Conclusions

We have successfully implemented the Josephus's problem mathematically in MATLAB by first keeping k constant and then taking recursive values of k to make the game more interesting. We have also implemented the same algorithm in choosing a same suit from a deck of 52 cards.

Reference

[1] R. Graham, D. Knuth, O. Patashnik, *Concrete Mathematics*, 2nd edition, Addison-Wesley, 1994.
[2] Robinson W.J.:\The Josephus Problem". Math. Gazette 44(1960), 47-52
[3] Woodhouse, D.:\The extended Josephus Problem". Rev.Mat.Hisp.-Amer (4)33 (1973), 207-218
[4] Armin Shams-Baragh, \Formulating The Extended Josephus Problem
[5] Lorez Halbeisen, ETH Zurich Norbert Hunger Buhler, University of Freiburg (Germany), The Josephus Problem
[6] Miguel A. Lerma, Josephus Problem.2003
[7] http://en.wikipedia.org/wiki/Josephus_problem

Appendix:

Code 1:

```
function [ output_args ] =test12Josephus(  )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
clc;

k=input('Enter k');
mat=input('Enter length of mat')
mat=1:1:mat;


tem=mat;
tem2=mat;
p=length(mat);
i=0;
kin=k;
while p>1

    t=mod((k-1),p)+1;
    w = waitforbuttonpress;
    if w==1
        disp('deleted element')
        tem(t)
    end

    tem=[tem(t+1:end) tem(1: t-1)] ;
    p=length(tem);

end


end
```

Code 2:

Code for controlling the Josephus Game GUI:

```
function varargout = JosephusProblemGame(varargin)
% JOSEPHUSPROBLEMGAME MATLAB code for JosephusProblemGame.fig
%      JOSEPHUSPROBLEMGAME, by itself, creates a new JOSEPHUSPROBLEMGAME or
raises the existing
%      singleton*.
%
%      H = JOSEPHUSPROBLEMGAME returns the handle to a new
JOSEPHUSPROBLEMGAME or the handle to
%      the existing singleton*.
%
%      JOSEPHUSPROBLEMGAME('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in JOSEPHUSPROBLEMGAME.M with the given input
arguments.
%
```

```
%      JOSEPHUSPROBLEMGAME('Property','Value',...) creates a new
JOSEPHUSPROBLEMGAME or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before JosephusProblemGame_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to JosephusProblemGame_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help JosephusProblemGame

% Last Modified by GUIDE v2.5 14-Nov-2014 22:20:57

% Begin initialization code - DO NOT EDIT

%Author : Shivam Sharma
%Student At Cluster Innovation Center , University of Delhi
%Copy Rights .All rights reserved by the author
%
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @JosephusProblemGame_OpeningFcn, ...
                   'gui_OutputFcn',  @JosephusProblemGame_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before JosephusProblemGame is made visible.
function JosephusProblemGame_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to JosephusProblemGame (see VARARGIN)

% Choose default command line output for JosephusProblemGame
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);
set(handles.edit1,'visible','on');
set(handles.edit2,'visible','on');
global StepsType Number_People Step
StepsType=0;
Number_People=0;
Step=0;
% UIWAIT makes JosephusProblemGame wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = JosephusProblemGame_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1
global StepsType Number_People Step



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double

set(handles.edit1,'visible','on');
global StepsType Number_People Step
Number_People=str2double(get(hObject,'String'));




% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
global StepsType Number_People Step
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2
global StepsType Number_People Step




% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3

global StepsType Number_People Step
%StepsType=get(hObject,'Value');




function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
set(handles.edit2,'visible','on');
global StepsType Number_People Step
Step=str2double(get(hObject,'String'));
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
global StepsType Number_People Step
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global StepsType Number_People Step choicePlay
if choicePlay==1
    delete(gcf);
    JosephusProblemGameDisplay_Shivam(Number_People,Step,StepsType)

elseif choicePlay==0
    s=test2(Number_People,Step,1);
    set(handles.edit7,'visible','on');
    set(handles.edit7,'String',[num2str(s) ' ']);
end


function out=test2(n,k,i)
global StepsType
    if StepsType==1
        if n==1
            out=1;

        else

            out=mod(test2(n-1,k,i+1)+k-1,n)+1;
        end
    elseif StepsType==2
        if n==1
            out=1;

        else

            out=mod(test2(n-1,k+1,i+1)+k-1,n)+1;
        end
    elseif StepsType==3
        if n==1
            out=1;
```

```
        else

            out=mod(test2(n-1,k,i+1)+(k*i)-1,n)+1;
        end
    end




% --- Executes when selected object is changed in uipanel2.
function uipanel2_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel2
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global StepsType Number_People Step
switch get(eventdata.NewValue,'Tag') %get tag of selected object
    case 'radiobutton1'
        StepsType=1;
    case 'radiobutton2'
        StepsType=2;
    case 'radiobutton3'
        StepsType=3;
    otherwise
        error('Sorry');
end









function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double
global StepsType Number_People Step

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
global StepsType Number_People Step
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when selected object is changed in uipanel7.
function uipanel7_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel7
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global StepsType Number_People Step  choicePlay
switch get(eventdata.NewValue,'Tag') %get tag of selected object
    case 'radiobutton11'
          set(handles.edit7,'Enable','off');
          set(handles.edit1,'Enable','on');
          set(handles.edit2,'Enable','on');
          choicePlay=1;
    case 'radiobutton12'
          set(handles.edit7,'Enable','on');
          choicePlay=0;
    otherwise
        error('Please Choose some value')

end

% --- Executes during object creation, after setting all properties.
function radiobutton11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to radiobutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
global StepsType Number_People Step

% --- Executes during object creation, after setting all properties.
function radiobutton12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to radiobutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
global StepsType Number_People Step
```

Code for playing the simulation:

```
function JosephusProblemGameDisplay_Shivam(number,kk,Type)

%Author : Shivam Sharma
%Student At Cluster Innovation Center , University of Delhi
```

```
%constants
global input1 steps angle r1 r a StepsType pos dat t
StepsType=Type;
input1=number;
steps=kk;
pos=0;
clc;
backColor = [0 0.7 0.7];

n=input1;
r=n+10;
a=pi/n;
s=0;
%create figure
figH = figure('position',[100 100 797 612], ...
              'menubar','none', ...
              'name','Akad Bakad Bambai Bo (A game Based on Josephus
Problem)', ...
              'numbertitle','off',...
              'color',backColor, ...
              'resize','on');


%create axes
axesH = axes('parent',figH, ...
             'units','pixels',...
             'visible','off','buttondownfcn',{@smallBDF},...
             'nextplot','add');




hold on
r1=0;

if n<20
    r1=4;
    fontsize=20;
    seq=8;
elseif n>=60||n<=100
    r1=4;
    fontsize=10;
    r=n+30;
    seq=10;
else
    r1=round(r/20);
    fontsize=10;
    seq=10;
end
```

```
%%
%menus
game = uimenu(figH,'label','Game');
uimenu(game,'label','New Game','Accelerator','n','callback',@newgame);
uimenu(game,'label','Rules','Accelerator','r','callback',@rules);
uimenu(game,'label','Solve Automatic','Accelerator','s','callback',@solve1);
uimenu(game,'label','Solve Manually','Accelerator','s','callback',@solve2);
uimenu(game,'label','Quit','Accelerator','q','callback','delete(gcf)');
help = uimenu(figH,'label','Help');
uimenu(help,'label','Help','callback',@menuHelp);
uimenu(help,'label','About','callback',@menuAbout);

angle=[];
for i=1:n
    s=s+2*a;
    angle(i)=s;
    viscircles([r*sin(s-2*a) r*cos(s-2*a)],r1,'EdgeColor','g');
     text((r)*sin(s-2*a),(r)*cos(s-2*a),...
     num2str(i),...
     'FontSize',fontsize,'HorizontalAlignment','center');

    drawnow;


     set(gca ,'DataAspectRatio',[1 1 1],'PlotBoxAspectRatio',[1 1 1])
end




% text((r+seq)*sin(s),(r+seq)*cos(s),...
%      'Josephus Problem',...
%      'FontSize',r,'HorizontalAlignment','center','Color',[1 0.8 0.4]);
%  set(gca ,'DataAspectRatio',[1 1 1],'PlotBoxAspectRatio',[1 1 1])



%%%%         %%
%%

 %The result is shown in the form of a table

freq=0;
probf=0;
figure('Position',get(0,'screensize'));
dat=cell(1,3);

%dat(1,:)={1,0,0};
columnname =   {'Serial.No', 'nth Person', 'Step'};
```

```
columnformat = {'numeric', 'numeric', 'numeric'};

columneditable =  [false false false];


t = uitable('Units','normalized','Position',...
            [0 0 10 1], 'Data', dat,...
            'ColumnName', columnname,...
            'ColumnFormat', columnformat,...
            'ColumnEditable', columneditable,...
            'RowName',[]);



%%

%setappdata
%setappdata(figH,'ph',ph);
setappdata(figH,'r',r);
%setappdata(figH,'cubeSize',cubeSize);
%setappdata(figH,'spaceSize',spaceSize);

%getappdata(figH)
end




function Positions(x,data1,data2)
global dat t
dat(x,:)={x,data1,data2};
%hTable: handle to the table
%newRow : the row of data you want to add to the table.
oldData = get(t,'Data');

newData = [oldData; dat(x,:)];

set(t,'Data',newData);
end


function solve1(obj,eventData)
global steps input1 angle r1 a StepsType
r = getappdata(gcf,'r');

    mat=1:1:input1;
    tem=mat;
    %tem2=mat;
    p=length(mat);

    pos=0;
   if  StepsType==1
```

```
            while p>1
                pos=pos+1;

                t=mod((steps-1),p)+1;

                pause(1)
                    viscircles([r*sin(angle(tem(t))-2*a) r*cos(angle(tem(t))-
2*a)],r1,'EdgeColor','r')
                    Positions(pos,tem(t),steps);

                tem=[tem(t+1:end) tem(1: t-1)] ;
                p=length(tem);
                r = getappdata(gcf,'r');
            end
   elseif StepsType==2
                while p>1
                        pos=pos+1;
                        t=mod((steps-1),p)+1;

                        pause(1)
                        viscircles([r*sin(angle(tem(t))-2*a)
r*cos(angle(tem(t))-2*a)],r1,'EdgeColor','r')
                        Positions(pos,tem(t),steps);
                        tem=[tem(t+1:end) tem(1: t-1)] ;
                        p=length(tem);
                        r = getappdata(gcf,'r');
                        steps=steps+1;
                 end

   elseif StepsType==3
                   l=1;
                   inn=steps;
                   while p>1
                       l=l+1;

                            t=mod((steps-1),p)+1;

                            pause(1)
                            viscircles([r*sin(angle(tem(t))-2*a)
r*cos(angle(tem(t))-2*a)],r1,'EdgeColor','r')
                           Positions(l,tem(t),steps);
                           tem=[tem(t+1:end) tem(1: t-1)] ;
                           p=length(tem);
                           r = getappdata(gcf,'r');
                           steps=inn*l;
                    end

   end



end
```

```
function solve2(obj,eventData)
global steps input1 angle r1 a StepsType pos
r = getappdata(gcf,'r');

    mat=1:1:input1;
    tem=mat;
    %tem2=mat;
    p=length(mat);

    pos=0;
    if StepsType==1
            while p>1
                pos=pos+1;
                t=mod((steps-1),p)+1;
                w = waitforbuttonpress;
                if w==1

                   viscircles([r*sin(angle(tem(t))-2*a) r*cos(angle(tem(t))-
2*a)],r1,'EdgeColor','r');
                   Positions(pos,tem(t),steps);
                end

                tem=[tem(t+1:end) tem(1: t-1)] ;
                p=length(tem);
                r = getappdata(gcf,'r');

            end
    elseif StepsType==2

        while p>1

                pos=pos+1;
                t=mod((steps-1),p)+1;
                w = waitforbuttonpress;
                if w==1

                    viscircles([r*sin(angle(tem(t))-2*a) r*cos(angle(tem(t))-
2*a)],r1,'EdgeColor','r');
                    Positions(pos,tem(t),steps);;

                end

                tem=[tem(t+1:end) tem(1: t-1)] ;
                p=length(tem);
                r = getappdata(gcf,'r');
                steps=steps+1;
         end

    elseif StepsType==3
        l=1;
        inn=steps;
        while p>1
            pos=pos+1;
                l=l+1;
                t=mod((steps-1),p)+1;
```

```
                w = waitforbuttonpress;
                if w==1

                    viscircles([r*sin(angle(tem(t))-2*a) r*cos(angle(tem(t))-
2*a)],r1,'EdgeColor','r');
                    Positions(pos,tem(t),steps);

                end

                tem=[tem(t+1:end) tem(1: t-1)] ;
                p=length(tem);
                r = getappdata(gcf,'r');
                steps=inn*l;
        end

    end




end

function newgame(obj,eventData)
close all;
delete(gcf);
JosephusProblemGame;
end
function menuAbout(obj,eventD)
s = ['Akad Bakad Bambai Bo(A game based on Joseph Problem) \copyright Shivam
Sharma,2014.', 10, ...
     'shiva.iuac@gmal.com', 10 10];
opt.WindowStyle = 'normal';
opt.Interpreter = 'tex';
warning off
msgbox(s,'About','none',opt);
warning on
end

function rules(obj,eventD)
s = ['This Game is based on Joseph problem in which people stand in a
circular arragement and every kth person is removed from the arrangement and
the counting starts from the next person '];
opt.WindowStyle = 'normal';
opt.Interpreter = 'tex';
warning off
msgbox(s,'About','none',opt);
warning on
end
function menuHelp(obj,eventD)
web flatRubiksHelp.html -helpbrowser
end
```