```
function varargout = lab6(varargin)
% LAB6 M-file for lab6.fig
%      LAB6, by itself, creates a new LAB6 or raises the existing
%      singleton*.
%
%      H = LAB6 returns the handle to a new LAB6 or the handle to
%      the existing singleton*.
%
%      LAB6('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LAB6.M with the given input arguments.
%
%      LAB6('Property','Value',...) creates a new LAB6 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before lab6_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to lab6_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 lab6

% Last Modified by GUIDE v2.5 29-Apr-2022 12:08:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @lab6_OpeningFcn, ...
                   'gui_OutputFcn',  @lab6_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 lab6 is made visible.
function lab6_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 lab6 (see VARARGIN)

% Choose default command line output for lab6
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

setup(handles)

% UIWAIT makes lab6 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function [task, args, anss] = description(task_num)
switch task_num
   case 1
      task = 'Площадь и периметр треугольника по сторонам'
      args = 'a b c'
      anss = 'S P'
   case 2
      task = 'Площадь и периметр ромба по диагоналям'
      args = 'd1 d2'
      anss = 'S P'
   case 3
      task = 'Площадь кольца по радиусам'
      args = 'r1 r2'
      anss = 'S'
   case 4
      task = 'Площадь и периметр квадрата по диагонали'
      args = 'd'
      anss = 'S P'
   case 5
      task = 'Расстояние между двумя точками. Площадь и периметр окружности с этим ↙
расстоянием в качестве радиуса.'
      args = 'x y a b'
      anss = 'r S P'
   case 6
      task = 'Площадь и периметр треугольника по двум сторонам и углу'
      args = 'a b A'
      anss = 'S P'
   case 7
      task = 'Площадь и периметр трапеции по сторонам и высоте'
      args = 'a b h'
      anss = 'S P'
   case 8
      task = 'Длина окружности и площадь круга по радиусу'
      args = 'R'
      anss = 'S L'

   otherwise
      warndlg('whaaat')
      [task, args, anss] = description_text(1)
end

function ok = check_args(args, task_num)
sizes = [3 2 2 1 4 3 3 1]

if size(args) ~= sizes(task_num)
```

```
    ok = 0
    warndlg('wrong number of arguments')
    return
end

if ( (task_num ~= 5) & (task_num ~= 6) ) & ( (args ~= abs(args)) | (prod(args) == 0)
)
    ok = 0
    warndlg('argument <= 0')
    return
end

ok = 1
switch task_num
    case 1
        [a,b,c] = deal(args(1), args(2), args(3))
        if (a+b) <= c || (b+c) <= a || (c+a) <= b
            ok = 0
            warndlg('impossible triangle')
        end
    case 6
        [a,b,A] = deal(args(1), args(2), args(3))

        if a <= 0 || b <= 0 || mod(A, 180) == 0
            ok = 0
            warndlg('impossible triangle')
        end
end

function draw_it(args, task_num)
switch task_num
    case 1
        [a, b, c] = deal(args(1), args(2), args(3))
        xy = [0 0 ; 0 a ; c 0 ; 0 0]
        plot(xy(:,1), xy(:,2))
    case 2
        args = args ./ 2
        [d1 d2] = deal(args(1), args(2))

        xy = [0 d1 ; d2 0 ; 0 -d1; -d2 0; 0 d1]
        plot(xy(:,1), xy(:,2))
    case 3
        [r1 r2] = deal(args(1), args(2))
        range = 0 : pi/20 : 2*pi

        plot(r1*cos(range), r1*sin(range), r2*cos(range), r2*sin(range))
    case 4
        d = args /2
        xy = [+d +d ; +d -d ; -d -d ; -d +d ; +d +d]
        plot(xy(:,1), xy(:,2))

    case 5
        [a b c d] = deal(args(1), args(2), args(3), args(4))
        range = 0 : pi/20 : 2*pi
```

```
        r = sqrt( abs(a-c)^2 + abs(b-d)^2 )

        plot(a + r*cos(range), b + r*sin(range))

    case 6
        [a b A] = deal(args(1), args(2), args(3))
        A = A * pi/180

        xy = [0 0 ; a 0 ; cos(A)*b sin(A)*b ; 0 0]
        plot(xy(:,1), xy(:,2))
    case 7
        [a b h] = deal(args(1), args(2), args(3))

        xy = [0 0 ; b 0 ; a h ; 0 h ; 0 0]
        plot(xy(:,1), xy(:,2))
    case 8
        r = args
        range = 0 : pi/20 : 2*pi

        plot(r*cos(range), r*sin(range))

end

function anss = handle_it(args, task_num)
switch task_num
    case 1
        [a,b,c] = deal(args(1), args(2), args(3))

        P = a+b+c
        p = P/2
        S = sqrt(p*(p-a)*(p-b)*(p-c))
        anss = [S, P]
    case 2
        [d1 d2] = deal(args(1), args(2))

        S = d1*d2/2
        P = 2*sqrt(d1^2 + d2^2)
        anss = [S P]
    case 3
        [r1, r2] = deal(args(1), args(2))
        anss = [pi* abs(r1^2-r2^2)]
    case 4
        anss = [ args^2/2 ]
    case 5
        [a b c d] = deal(args(1), args(2), args(3), args(4))

        r = sqrt( abs(a-c)^2 + abs(b-d)^2 )
        S = pi*r^2
        P = 2*pi*r
        anss = [r S P]
    case 6
        [a b A] = deal(args(1), args(2), args(3))
        A = A * pi/180

        S = a*b*sin(A)/2
```

```
        P = a+b+ sqrt(a^2+b^2-2*a*b*cos(A))
        ans = [S P]
    case 7
        [a b h] = deal(args(1), args(2), args(3))

        S = h*(a+b)/2
        P = a+b+h + sqrt(h^2+(a-b)^2)
        ans = [S P]
    case 8
        [r] = deal(args)

        S = pi*r^2
        P = 2*pi*r
        anss = [S P]

end

function do_stuffs(handles)
global Task_number
args = str2num(get(handles.edit1, 'string'))

reset_axes(handles)

if check_args(args, Task_number) == 0
    return
end

draw_it(args, Task_number)
change_grid(handles, get(handles.radiobutton1, 'value'))

set(handles.text4, 'string', handle_it(args, Task_number))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function setup(handles)
reset(handles)
change_task(handles, 1)

global Task_number

names = []
for i = 1:8
    names = [names ; sprintf('Task%d', i)]
end

set(handles.popupmenu1, 'string', names)

set(handles.radiobutton1, 'string', 'grid')
set(handles.pushbutton1, 'string', 'do it')
set(handles.pushbutton2, 'string', 'reset')

function reset(handles)
reset_axes(handles)

set(handles.edit1, 'string', '')
```

```
function reset_axes(handles)
cla reset;
%set(handles.radiobutton1, 'value', 0)
%change_grid(handles, 0)
change_grid(handles, get(handles.radiobutton1, 'value'))
set(handles.text4, 'string', '')

function change_task(handles, task_num)
reset(handles)

global Task_number
Task_number = task_num

[task, args, anss] = description(task_num)
set(handles.text1, 'string', task)
set(handles.text2, 'string', args)
set(handles.text3, 'string', transpose(anss(1:2:end)))

function change_grid(handles, enabled)
if enabled
    set(handles.axes1, 'XGrid', 'on')
    set(handles.axes1, 'YGrid', 'on')
else
    set(handles.axes1, 'XGrid', 'off')
    set(handles.axes1, 'YGrid', 'off')
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)
do_stuffs(handles)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
reset(handles)

% --- 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
change_grid(handles, get(hObject,'Value'))

% --- Executes on selection change in popupmenu1.
```

```
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
change_task(handles, get(hObject,'Value'))


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Outputs from this function are returned to the command line.
function varargout = lab6_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 during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

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



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


% --- 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.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```