PROBLEM 1. See the comments for more explanation.

- (a)

```
% Homework 6
% Problem 1

% deletes all current figures.
close all;

% removes all variables from the workspace.
clc;

N = 2^20;

% starts a stopwatch timer.
tic;

% compute DFT of a random vector using FFT.
x=fft(rand(1,N));

% prints the elapsed time since tic was used.
toc
```

- (b)

```
function out_vector=MYDFT(in_vector)

%-----------------------------------
% Computation of DFT using for.
%-----------------------------------

L = length(in_vector);

for k=1:L
```

```
out_vector(k) = 0;

for j=1:L
temp = in_vector(j) * exp(-i*2*pi*(j-1)*(k-1)/L);
out_vector(k) = out_vector(k) + temp;
end

end

%-----------------------------------
```

- (c)

```
function out_vector=MYmodDFT(in_vector)

%-----------------------------------
% Computation of DFT using butterfly method.
%-----------------------------------

L=length(in_vector);

% add a zero entry if the length of in_vector is odd
if (mod(L,2)~=0)
input=zeros(1,L+1);
input(1:end-1)=in_vector;
in_vector=input;
L=L+1;
end

% separate odd and even parts
in_even=in_vector(1:2:end-1);
in_odd=in_vector(2:2:end);

% take DFT from both parts
out_even=MYDFT(in_even);
out_odd=MYDFT(in_odd);

% compute out-vector considering correct weights
W=exp(-i*2*pi/L);
```

```
weights=W.^(0:(L/2)-1); % half of the weights are sufficient.

out_vector_1 = out_even + weights .* out_odd;
out_vector_2 = out_even + W^(L/2)* weights .* out_odd;
out_vector=[out_vector_1, out_vector_2];

%-----------------------------------
```

- (d)

  It is a program for comparing elapsed time for taking DFT using different methods

```
% Homework 6
% Problem 1

close all;
clc;

N= 9:13 ;
t=zeros(3,5);

for i = 9:13

x=rand(2^i,1);

tic;
X=fft(x);
t(1,i-8) = toc

tic;
X=MYDFT(x);
t(2,i-8) = toc

tic;
X=MymodDFT(x);
t(3,i-8) = toc

end

hold on;
```

```
plot(t(1,:),'r')
plot(t(2,:),'b')
plot(t(3,:),'g')
```

PROBLEM 2. `conv(x,h)` in MATLAB uses DFT in order to convolve x and h linearly. In fact, it uses `fft` to compute DFT of x and h and then multiplies their DFTs and finally takes the inverse DFT to obtain result (recall: convolution in time domain is equivalent to multiplication in frequency domain).

In order to compare elapsed time for running convolution operation in different methods we should write down convolution formula without using `conv`. Since it is not so interesting, we just focus on theoretical results instead of time comparison.

In this problem you are be familiar with overlap-add and overlap-save methods in order to compute linear convolution more easily than direct method when the length of input signal is large.

- (a)

```
% L is length of input signal and P is length of impulse response.
L=100000; P=20;
x=rand(1,L);
h=100./((0:P-1)+13);

% Direct convolution
tic;
y1=conv(x,h);
toc
```

- (b)

$$\begin{aligned} y[n] &= x[n] * h[n] \\ &= (\sum_{r=0}^{+\infty} x_r[n - rB]) * h[n] \\ &= \sum_{r=0}^{+\infty} (x_r[n - rB] * h[n]) \\ &= \sum_{r=0}^{+\infty} y_r[n - rB] \end{aligned}$$

- (c)

```
% overlap-add method :  "add" in output(!)
```

```
y2=zeros(size(y1));
temp=zeros(size(y1));
B=50;

tic;
for i=1:(L/B)

% each time we consider a B-points window of x, convolve it with h and
% save the output in B+(P-1) points of temp
temp( (i-1)*B + 1 :  i*B + (P-1) )=conv(x( (i-1)*B + 1 :  i*B ),h);

% add with previous results considering overlaps
y2=y2+temp;

% make temp zero
temp=zeros(size(y1));

end
toc
```

- (d) Assume that y_1 is the output of linear convolution between B-points signal x and P-points impulse response h (where $P < B$). So, length of y_1 is $B + P - 1$. We also know that the output of circular convolution of x and h has length B (it must have the same length as input signal x). If we call this signal y_2, we will have:

$$\begin{aligned} y_2[0] &= y_1[0] + y_1[B] \\ y_2[1] &= y_1[1] + y_1[B+1] \\ &\dots \\ y_2[P-2] &= y_1[P-2] + y_1[B+P-2] \\ &\textit{and} \\ y_2[P-1] &= y_1[P-1] \\ &\dots \\ y_2[B-1] &= y_1[B-1] \end{aligned}$$

  So, $y_2[n] = y_1[n]$ except for first $P - 1$ indices.

- (e)

$$\begin{aligned} x_r[n] &= x[n + r(B - P + 1) - P + 1] \quad 0 \le n \le B - 1 \\ y[n] &= \sum_{r=0}^{+\infty} y_r[n - r(B - P + 1) + P - 1] \end{aligned}$$

where

$$\begin{cases} y_r[n] = y_{rp}[n] & P-1 \leq n \leq B-1 \\ y_r[n] = 0 & otherwise \end{cases}$$

and $y_{rp}[n]$ is the circular convolution of $x_r[n]$ with $h[n]$.

- (f)

```
% overlap-save method:  "save" in input(!)

% add P-1 extra zeros at input
extra_zeros=zeros(1,P-1);
x=[extra_zeros, x];
L=length(x);
y3=[];

tic
while(length(x)>=B)

% circular convolution using cconv; recall:  cconv(x,h,length(x))
% returns the output of filter h when input is x.
% Lengths of input and output are equal to each other.

% circular convolve for one B-points window from x
temp3=cconv(x(1:B),h,B);

% save results without considering first P-1 points
y3=[y3, temp3(P:end)];

% shift window to the right considering P-1 points overlapped
% with preceding section.
x=x( ((B+1)-(P-1)) :  end);

end
toc
```