

```

#!/perl
#
# This program is a limited implementation of the algorithm given by Dr. Yu
# on dynamic programming.
#
# Array Definitions
# Put initial manual calculations in the array
# This includes v1 - v4(x) calculations done manually
# This program can be modified to generate these arrays based on the Problem inputs
# but, in this example these arrays are fed manually using the following arrays
#
@capx = (0,10,20,30,40,50,60,70,80,90,100);
@v1x = (0,8,16,16,16,16,16,16,16,16);
@v2x = (0,0,13,13,26,26,39,39,39,39,39);
@v3x = (0,0,0,20,20,20,40,40,40,40,40);
@v4x = (0,0,0,0,30,30,30,30,30,30,30);

#Initiation: Note that V1*(x) is nothing but V1(x)
@v1starx = @v1x;

# Call the function which generates the column for V2*(x), given V1*(x)
# and also returns the location of the maximum profit with row number
@v2starx_with_location = &starx_finder (\@v1starx,\@v2x);

#
$countlength = @v2starx_with_location/2 - 1;
for ($count=0; $count<=$countlength; $count++){
$v2starx[$count] = $v2starx_with_location[$count*2];
$y2starx[$count] = $capx[$v2starx_with_location[$count*2+1]];
$x2starx[$count] = $capx[$count] - $y2starx[$count];
}

# Call the function which generates the column for V3*(x), given V2*(x)
# and also returns the location of the maximum profit with row number
@v3starx_with_location = &starx_finder (\@v2starx,\@v3x);
$countlength = @v3starx_with_location/2 - 1;
for ($count=0; $count<=$countlength; $count++){
$v3starx[$count] = $v3starx_with_location[$count*2];
$y3starx[$count] = $capx[$v3starx_with_location[$count*2+1]];
$x3starx[$count] = $capx[$count] - $y3starx[$count];
}

```

```

# Call the function which generates the column for V4*(x), given V3*(x)
# and also returns the location of the maximum profit with row number
@v4starx_with_location = &starx_finder (\@v3starx,\@v4x);
$countlength = @v4starx_with_location/2 - 1;
for ($count=0; $count<=$countlength; $count++){
$v4starx[$count] = $v4starx_with_location[$count*2];
$y4starx[$count] = $capx[$v4starx_with_location[$count*2+1]];
$x4starx[$count] = $capx[$count] - $y4starx[$count];
}

```

**# Output the generated data in a CSV format so that it can be opened by Excel**

```

print "\nx,V1(x),V2(x),V3(x),V4(x),V2*(x),y2*(x),x2*(x),V3*(x),y3*(x),x3*(x),V4*(x),y4*(x),x4*(x)
\n";
$countlength=@capx-1;
for ($count=0; $count<=$countlength; $count++){
print "$capx[$count],$v1x[$count],$v2x[$count],$v3x[$count],$v4x[$count],$v2starx[$count],
$y2starx[$count],$x2starx[$count],$v3starx[$count],$y3starx[$count],$x3starx[$count],$v4starx
[$count],$y4starx[$count],$x4starx[$count]\n";
}

```

**###**

**# Implementation of algorithm to find  $V_i^*(x)$**

```

sub starx_finder {
my ($A1, $A2) = @_ ;
my @array1 = @{$A1};
my @array2 = @{$A2};
my $arraylength=@array1-1;
my @tmparray;
for ( $countA=0; $countA <= $arraylength ; ++$countA ){
for ( $countB=0; $countB <= $countA; ++$countB ){
$tmparray[$countB] = $array1[$countB]+$array2[$countA-$countB];
}
}
$max_value_array = &give_me_max (@tmparray);
$arrayout[$countA*2] = $max_value_array[0];
$arrayout[$countA*2+1] = $max_value_array[1];
}
return @arrayout;
}

```

**# Function to find the location of the maximum for each iteration of calculation**

```

sub give_me_max {

```

```
my @inputarray = @_;  
my @sortedarray = sort {$a <=> $b} @inputarray;  
my $max = $sortedarray[@sortedarray - 1];  
my $max_location = 0;  
my $arraylength=@sortedarray-1;  
LOOP1: for ( my $sub_count=0; $sub_count <= $arraylength; $sub_count++){  
if ($inputarray[$sub_count] == $max ){  
$max_location = $sub_count;  
last LOOP1;  
}  
}  
return $max, $max_location;  
}
```