THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP343
Fundamentals of Multimedia Computing

Spring 2000

Final Examination

Tuesday, 23 May 2000
12:30 - 1:40pm (1 hour 10 minutes)

This is version 1 of the exam.

Instructions

- This is a closed-book, closed-notes examination
- Simple calculators are permitted; PDAs and computers are not
- Question 1 identifies which exam version you are using
- Questions 2-11 have 2 marks each, and questions 12-27 have 5 marks each
- The last page has been printed in colour.
- Answer all the questions on the multiple choice answer sheet, which you need to return for marking
- Clearly write your name, student number and lab section on the answer sheet, and mark the digits of your student id on the sheet

Question 1
What version of the exam are you using? This is written on the front.

a) Version 1
b) Version 2

[2 marks each for Questions 2 – 11. Some questions have three possible answers, and some questions have two possible answers.]

Question 2 Which one of the following would require the least processing?

a) The negation of a 100*100, 8 bit CLUT image
b) The negation of a 20*20, 24 bits per pixel RGB image
c) Applying blur over all pixels in a 80*80, 24 bits per pixel RGB image

Question 4 One of the following two recordings has more quantisation error (on average) than the other one, otherwise it is the same.

Which one has the biggest level of quantisation error?

a) A.wav

b) B.wav

Question 5 Which one of the following recordings has the most sine waves of different frequencies? (Both recordings last for the same length of time).

a) a.wav

b) b.wav

Question 6 Someone is thinking of different ways to write a program to create a single summary image that represents a video sequence. Which one of the following would be *least* helpful?

a)

$$\frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}|C(x+k,y+l)-R(x+i+k,y+j+l)|$$

b)

```
if  (this_byte & 0x0F == 0x06) . . .
```

[This is C code]

c)

$$\sqrt{\sum_{j=1}^{M}(This(j)-That(j,r))^2}$$

Question 7 A program loads a soundfile into memory. It then makes a new soundfile using the following procedure:

- output sample X_i
- output sample X_{i+1}
- completely ignore sample X_{i+2}
- i=i+3
- then repeat (back to the top)

This pattern continues until all samples in the input soundfile have been processed.

If the original soundfile was a recording of a 600Hz triangle wave, the output soundfile will be a 200Hz triangle wave. True or false?

a) True
b) False

Question 8 Which one of the following kernels would appear to create the most obvious edge effect?

a)

-1/9	-1/9	-1/9
-1/9	+17/9	-1/9
-1/9	-1/9	-1/9

b)

0	-1	0
-1	4	-1
0	-1	0

Question 9 An RGB colour cube is created with each axis having a range from 0 to 1. A CMY colour cube is created with each axis also having a range from 0 to 1. The colour at point (0.5,0.5,0.5) will be the same for both. True or false?

a) True
b) False

Question 10 Here are three formats. Two of them have a very similar approach. In contrast, the other one can handle many extra techniques – which one is this?

a) MPEG 1
b) MPEG 2
c) MPEG 4

Question 11 Someone is starting to write a program to help work out what format an input file is. To do this, the program takes the first 16 bytes of the file. Then, for each byte,

- if the byte is not within the range of ASCII printable characters, the byte is ignored (nothing printed).
- if the byte is within the range of ASCII printable characters, it is printed.

The program is run and the following message is printed.

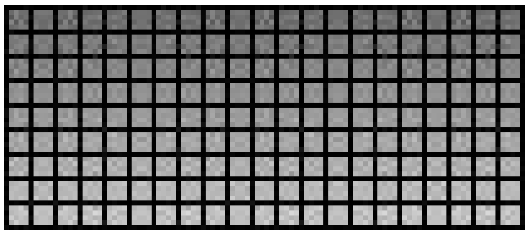
```
.sndV$
```

What format is the file?

a) WAV
b) AU
c) PPM

[5 marks each for Questions 12 – 27. All questions have five possible answers.]

Q13)
Here is a 24 bits per pixel image. Black lines are used *only* to indicate the



individual pixels for the sake of the exam - those lines are not actually visible in the image and would not get stored, etc. Assume that all pixels are identical and have a level of gray which is half that of white.

The image is saved in ppm ‘P3’ format using the filename gray_p3.ppm, and in ppm ‘P6’ format, using the filename gray_p6.ppm. The file sizes can then be compared.

In terms of the file size, what percentage of gray_p3.ppm is gray_p6.ppm? For example, if you think gray_p6.ppm is 10kB and gray_p3.ppm is 100kB, your answer would be 10%.

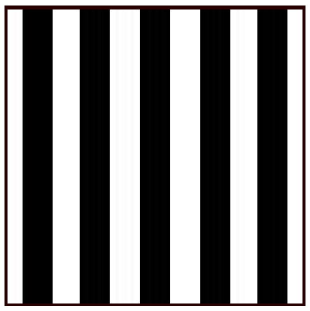
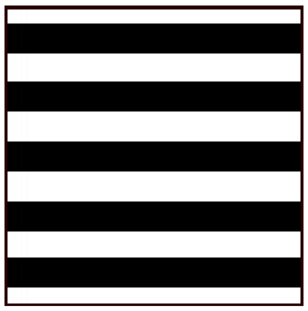
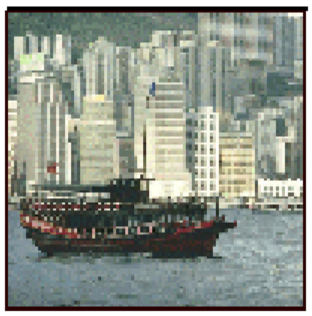
To help you answer this question, the output from ‘man ppm’ is shown below. Assume that the minimum amount of whitespace is used. Possible answers are shown later.

```
NAME
     ppm - portable pixmap file format

DESCRIPTION
     The portable pixmap format is a  lowest  common  denominator
     color image file format.  The definition is as follows:

     - A "magic number" for identifying the  file  type.   A  ppm
       file's magic number is the two characters "P3".

     - Whitespace (blanks, TABs, CRs, LFs).

     - A width, formatted as ASCII characters in decimal.

     - Whitespace.

     - A height, again in ASCII decimal.

     - Whitespace.

     - The maximum color-component value, again in ASCII decimal.

     - Whitespace.

     - Width * height pixels, each  three  ASCII  decimal  values
       between 0 and the specified maximum value, starting at the
       top-left  corner  of  the  pixmap,  proceeding  in  normal
       English  reading  order.   The three values for each pixel
       represent red, green, and blue, respectively; a value of 0
       means  that color is off, and the maximum value means that
       color is maxxed out.

     - Characters from a "#" to the next end-of-line are  ignored
       (comments).

     - No line should be longer than 70 characters.

     Here is an example of a small pixmap in this format:
     P3
     # feep.ppm
     4 4
     15
      0  0  0    0  0  0    0  0  0   15  0 15
      0  0  0    0 15  7    0  0  0    0  0  0
      0  0  0    0  0  0    0 15  7    0  0  0
     15  0 15    0  0  0    0  0  0    0  0  0

     Programs that read this format should be as lenient as  pos-
     sible, accepting anything that looks remotely like a pixmap.

     There is also a variant on the format, available by  setting
     the  RAWBITS  option  at  compile  time.   This  variant  is
     different in the following ways:

     - The "magic number" is "P6" instead of "P3".
     - The pixel values are stored as  plain  bytes,  instead  of
       ASCII decimal.

     - Whitespace is not allowed in the pixels area, and  only  a
       single  character  of  whitespace (typically a newline) is
       allowed after the maxval.

     - The files are smaller and many times faster  to  read  and
       write.

     Note that this raw format can only be used for maxvals  less
     than or equal to 255.
```

The possible answers are shown below. Choose the closest answer.

a) 15%
b) 30%
c) 45%
d) 60%
e) 75%

Q14)
At *http://www.envisiondev.com/internetdev/oct96_w/wg_comp.htm* there is a web page which talks about Gif compression. Three images shown below are scanned and saved in GIF format (each one 300 * 300 pixels).

Someone is trying to work out which image has the *smallest* file size. Use your knowledge of how GIF works to state which one of the following statements is completely true.

a) The left image has a file size which is smaller than the other two
b) The middle image has a file size which is smaller than the other two
c) The right image has a file size which is smaller than the other two
d) The left image and the middle image have the same file size, and that size is smaller than the right image
e) All three images have the same file size

Q16)
Which one of the following is *not* a metafile format as described on the course?

a) Microsoft Word
b) Microsoft Excel
c) Postscript
d) TIFF
e) Flash

Q17)
A sequence of MIDI bytes is sent to a MIDI synthesizer. When they are processed by the MIDI synthesizer they sound like this: some_notes

Assume that the following type of byte sequence is used to stop a note:

`1001 cccc`	`0 nnnnnnn`	`0 0000000`

How many of those MIDI bytes contain velocity information (of *any* velocity value)?
a) 2
b) 3
c) 4
d) 5
e) 6

Q18)
Here is some extracts from the course notes about MIDI:

- If you play a note on a channel, you will hear it with the default channel sound (for example, the default sound for channel 0 may be a piano)
- To change the default sounds, you need to issue a *program change* command, as follows:

Byte 1	*Byte 2*
`1100 cccc`	`0 ppppppp`
Status byte *(1101= Program change)* *(c=channel)*	*Data byte 1* *(p=program, meaning the number of the sound)*

Part of the general midi tables:

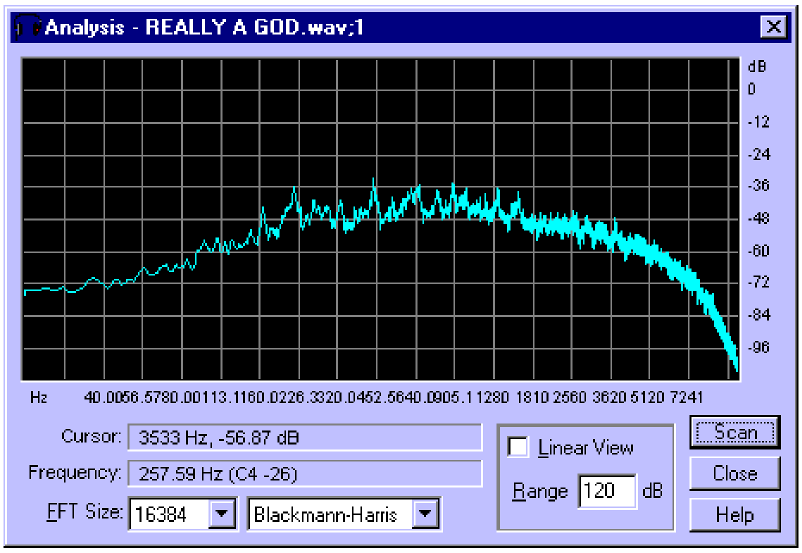
Piano	Chromatic Percussion	Organ
0. Acoustic Grand Piano	8. Celesta	16. Hammond Organ
1. Bright Acoustic Piano	9. Glockenspiel	17. Percussive Organ
2. Electric Grand Piano	10. Music Box	18. Rock Organ
3. Honky-tonk Piano	11. Vibraphone	19. Church Organ
4. Rhodes Piano	12. Marimba	20. Reed Organ
5. Chorused Piano	13. Xylophone	21. Accordion
6. Harpsichord	14. Tubular Bells	22. Harmonica
7. Clarinet	15. Dulcimer	23. Tango Accordion

Guitar	Bass
24. Acoustic Guitar (nylon)	32. Acoustic Bass
25. Acoustic Guitar (steel)	33. Electric Bass (finger)
26. Electric Guitar (jazz)	34. Electric Bass (pick)
27. Electric Guitar (clean)	35. Fretless Bass
28. Electric Guitar (muted)	36. Slap Bass 1
29. Overdriven Guitar	37. Slap Bass 2
30. Distortion Guitar	38. Synth Bass 1
31. Guitar Harmonics	39. Synth Bass 2

Assume that every MIDI channel plays the Acoustical Grand Piano (general MIDI instrument number 0) by default. If the bytes 1100 0001 0001 1111 are sent to a MIDI synthesiser and then a note is played on the highest MIDI channel, the note timbre will be from an instrument in which group?

a) An instrument in the Piano group
b) An instrument in the Chromatic Percussion group
c) An instrument in the Organ group
d) An instrument in the Guitar group
e) An instrument in the Bass group

Q19)
An audio file contains three seconds of speech. The complete soundfile (16 bits per sample) is analysed using CoolEdit. The following display is generated.



Analysis - REALLY A GOD.wav;1

Cursor: 3533 Hz, -56.87 dB

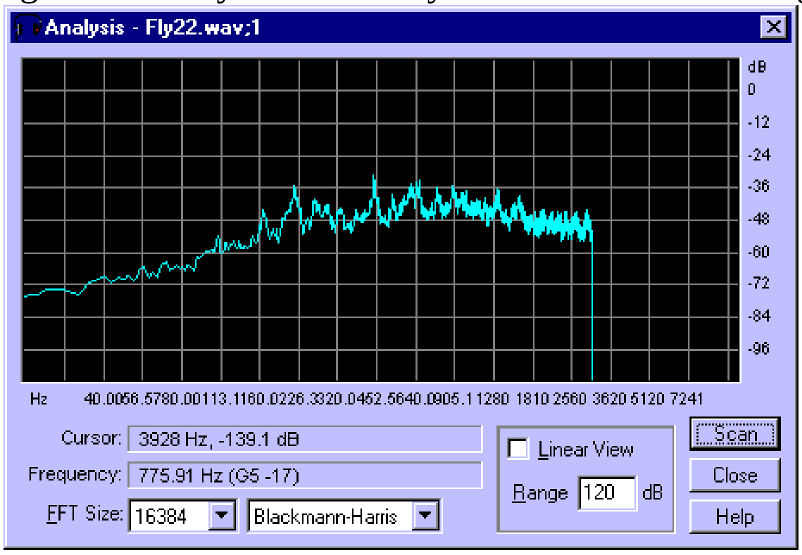
Frequency: 257.59 Hz (C4 -26)

FFT Size: 16384 Blackmann-Harris

☐ Linear View

Range 120 dB

Some processing then takes place. The complete soundfile is then analysed again in exactly the same way as before. The following display is generated.



Analysis - Fly22.wav;1

Cursor: 3928 Hz, -139.1 dB

Frequency: 775.91 Hz (G5 -17)

FFT Size: 16384 Blackmann-Harris

☐ Linear View

Range 120 dB

What processing took place between the two ?

a) Every other sample was deleted (for each pair of samples $X_{i,}$ X_{i+1} -> X_i)
b) Two copies were made of every sample (X_i -> X_i, X_i)
c) The samples are low pass filtered at 3000Hz.
d) Samples were converted into 1 byte per sample. Then they were converted into 2 bytes per sample.
e) Chorus was applied to the soundfile

Q21)

Macromedia Director has a special scripting language which developers can use to develop intelligent multimedia applications. What is the name of the scripting language?

a) Keykit
b) Perl
c) Authorware
d) Lingo
e) Orchestrator Pro

Q23)
Which is the hardest to achieve?

a) Conversion from vector to bitmap format
b) Conversion from bitmap to vector format
c) Resizing a vector image
d) Resizing a bitmap image
e) Changing the colour of all lines stored in vector format

Q24)
Stereo positioning is the adjustment of the relative balance of different sound channels which affects the position of the perceived sound within the stereo field. If:

- -1 <= $stereo_position <= 1
- a value of -1 would mean that the right channel would be silent and the left channel doubled
- a value of 1 would mean that the left channel would be silent and the right channel doubled

then which one of the following can correctly perform stereo positioning, and only stereo positioning, on a 16-bit stereo file?

a)

```
for ($i = 0; $i < $datalength/4; $i++)
{
        read (INPUT, $left, 2, 0);
        read (INPUT, $right, 2, 0);

        $left = ($left - $left * $stereo_position) * 4 * $i/$datalength;
        $right = ($right + $right * $stereo_position) * 4 * $i/$datalength;

        print OUTTEMP $left;
        print OUTTEMP $right;
        $outdatalen += 4;
    }
```

b)

```
    for ($i = 0; $i < $datalength/2; $i++)
    {
        read (INPUT, $left, 1, 0);
        read (INPUT, $right, 1, 0);

        $left = $left + $left * $stereo_position;
        $right = $right - $right * $stereo_position;

        print OUTTEMP $left;
        print OUTTEMP $right;

        $outdatalen += 2;
    }
```

c)

```
for ($i = 0; $i < $datalength/4; $i++)
{
        read (INPUT, $left, 2, 0);
        read (INPUT, $right, 2, 0);

        $left = $left - ($left * $stereo_position);
        $right = $right + ($right * $stereo_position);

        print OUTTEMP $left;
        print OUTTEMP $right;

        $outdatalen += 4;
}
```

d)

```
for ($i = 0; $i < $datalength/2; $i++)
{
        read (INPUT, $left, 1, 0);
        read (INPUT, $right, 1, 0);

        $left = ($left + $right)/2 * $stereo_position;
        $right = ($left + $right)/2 * (1/$stereo_position);

        print OUTTEMP $left;
        print OUTTEMP $right;

        $outdatalen += 2;
}
```

e)

```
for ($i = 0; $i < $datalength/4; $i++)
{
        read (INPUT, $left, 2, 0);
        read (INPUT, $right, 2, 0);

        $left = ($left + $right)/2 * $stereo_position;
        $right = ($left + $right)/2 * (1/$stereo_position);

        print OUTTEMP $left;
        print OUTTEMP $right;

        $outdatalen += 4;
}
```

Q25)
Which of images (a) to (e) is generated from the code shown following, given the input images shown later?

```
#!/usr/local/bin/perl5

use GD;

open(Input,"input.png")||die;
$Image=newFromPng GD::Image(Input)||die;
close Input;

open(Input,"input.png")||die;
$Src=newFromPng GD::Image(Input)||die;
close Input;

@index=$Image->getBounds;
$width = @index[0];
$height = @index[1];


$i=0;
while($i<$width)
{
    $oldy = $height/2; $newy = $height/2+20*sin($i/20);
    $Image->copyResized($Src, $i,0,$i,0,1,$newy,1,$oldy);
    $Image->copyResized($Src,$i,$newy,$i,$oldy,1,$height-$newy,
                                                  1,$height-$oldy);
    $i++;
}

open (output, ">output.png")||die;
binmode output;
print output $Image->png;
close output;
```

Note that most computer languages, Perl included, use radians and not degrees. There are 2*pi radians in a circle. So, for example, the following code:

```
$pi=3.14159;

for ($angle=0; $angle<=(2*$pi);  $angle+=($pi/2) ) {
        printf "%1.5f %1.3f\n", $angle, sin($angle);  }
```

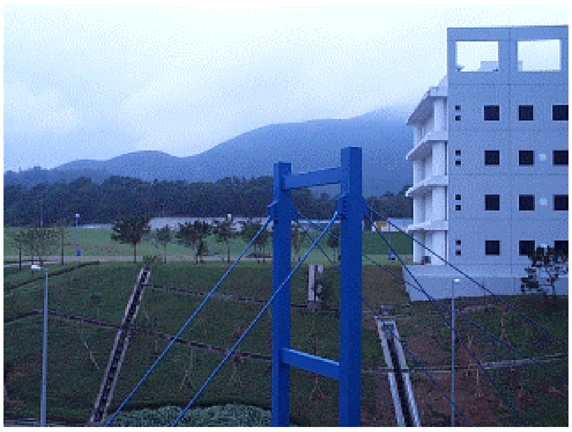
produces this:

```
0.00000 0.000
1.57079 1.000
3.14159 0.000
4.71238 -1.000
6.28318 0.000
```

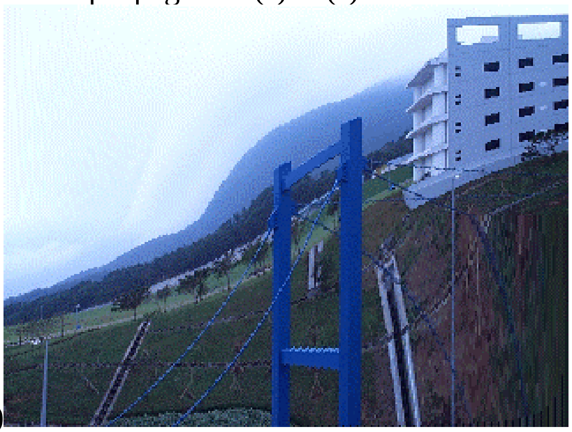
It may also be useful to remind you of the parameters used by copyResized:

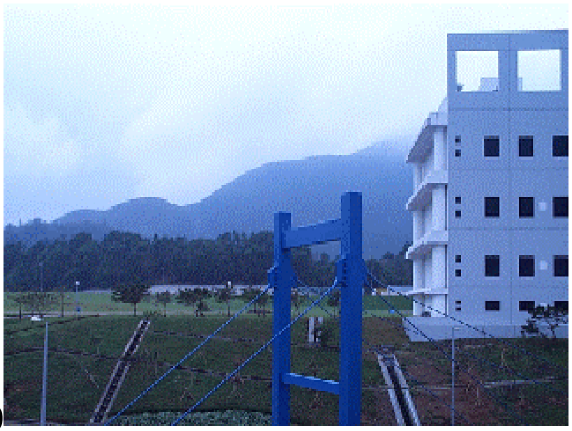
```
$image->copyResized(sourceImage,dstX,dstY,srcX,srcY,destW,destH,srcW,srcH)
```

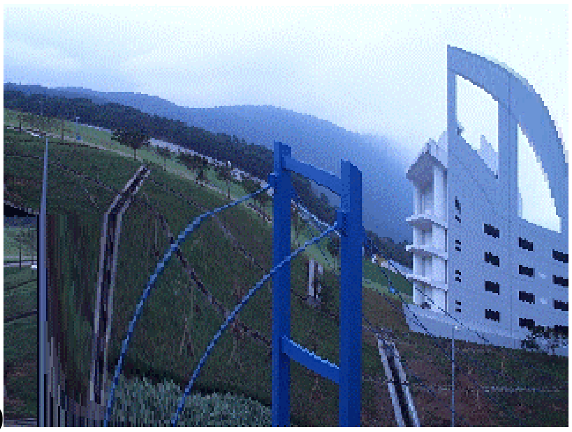
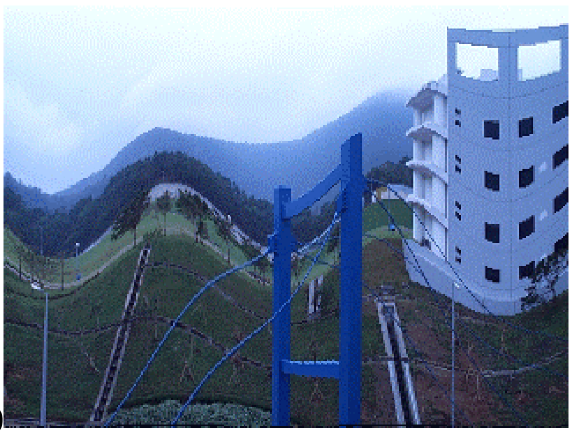
“The source and destination rectangle's are specified independently by (srcW,srcH) and (destW,destH) respectively.”

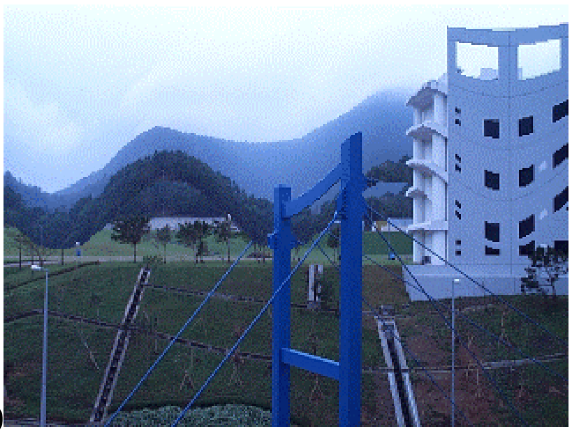
input.png: 

Select output.png from (a) to (e) as follows.

(a) 

(b) 

(c)  (d)

(e)

Q26)
On the following pages are five programs. Each program begins with the following code, which is shown here to save space:

```
#!/usr/local/bin/perl5

# This is the initial code for each program

use GD;

open(Input,"input.png")||die;
$Image=newFromPng GD::Image(Input)||die;
close Input;
```

(a)

```
# Initial code goes here

@index=$Image->getBounds;
$width = @index[0];
$height = @index[1];

for ($j=0; $j<$height; $j++)
{
    for ($i=0; $i<$width; $i++)
    {
        $data = $Image->getPixel($i,$j);
        $ImageBuf[$i][$j] = $data;
    }
}

for ($j=0; $j<$height; $j++)
{
    for ($i=0; $i<$width/2; $i++)
    {
     $pixel1 = $Image->getPixel($i,$j);
     $pixel2 = $Image->getPixel($width-$i-1, $j);
     $Image->setPixel($width-$i-1, $j,$pixel1);
     $Image->setPixel($i, $j,$pixel2);
    }
}

open (output, ">output.png")||die;
binmode output;
print output $Image->png;
close output;
```

(b)

```
# Initial code goes here

for ($j=0; $j<256; $j++)
{
 ($r,$g,$b)=$Image->rgb($j);
 &SetRGB($Image, $j, (int($r/32)*32), (int($g/32)*32),
(int($b/32)*32));
}

open (output, ">output.png")||die;
binmode output;
print output $Image->png;
close output;

sub SetRGB # set the $colorindex in CLUT to be $newr, $newg, $newb
{
 local($Image,$colorindex,$newr,$newg,$newb)=@_;
 $Image->colorDeallocate($colorindex);
 do
 {
  $newindex = $Image->colorAllocate($newr,$newg,$newb);
 } until ($colorindex==$newindex);
}
```

(c)

```
# Initial code goes here

for ($j=0; $j<256; $j++)
{
 ($r,$g,$b)=$Image->rgb($j);
 $temp1 = ($r+$g+$b)/3;
 if($temp1+30>255)
 {
   $temp2 = 255;
 }
 else
 {
   $temp2 = $temp1+30;
 }
 &SetRGB($Image, $j,$temp2, $temp2, $temp1);
}

open (output, ">output.png")||die;
binmode output;
print output $Image->png;
close output;

sub SetRGB # set the $colorindex in CLUT to be $newr, $newg, $newb
{
 local($Image,$colorindex,$newr,$newg,$newb)=@_;
 $Image->colorDeallocate($colorindex);
 do
 {
  $newindex = $Image->colorAllocate($newr,$newg,$newb);
 } until ($colorindex==$newindex);
}
```

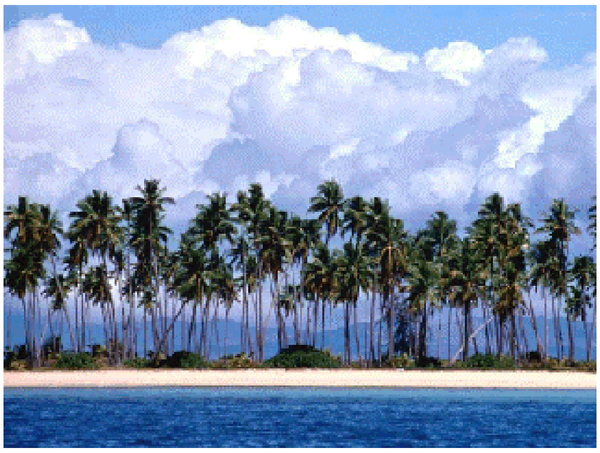
(d)

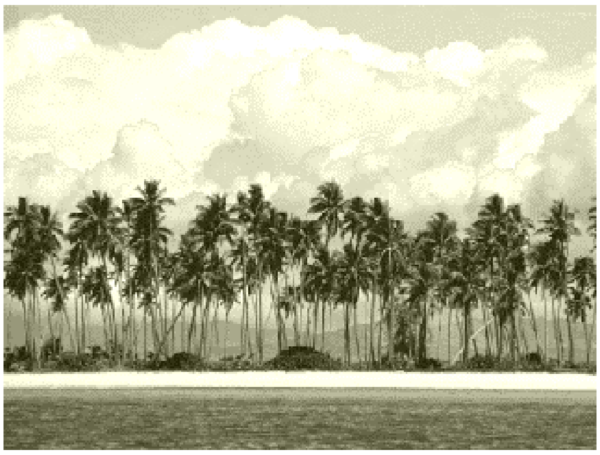
```
# Initial code goes here

for ($j=0; $j<256; $j++)
{
 ($r,$g,$b)=$Image->rgb($j);
 &SetRGB($Image, $j, 255-$r, 255-$g, 255-$b);
}

open (output, ">output.png")||die;
binmode output;
print output $Image->png;
close output;

sub SetRGB # set the $colorindex in CLUT to be $newr, $newg, $newb
{
 local($Image,$colorindex,$newr,$newg,$newb)=@_;
 $Image->colorDeallocate($colorindex);
 do
 {
  $newindex = $Image->colorAllocate($newr,$newg,$newb);
 } until ($colorindex==$newindex);
}
```

(e)

```
# Initial code goes here

for ($j=0; $j<256; $j++)
{
 ($r,$g,$b)=$Image->rgb($j);
 &SetRGB($Image, $j, (int($r/2)*2), (int($g/2)*2), (int($b/2)*2));
}

open (output, ">output.png")||die;
binmode output;
print output $Image->png;
close output;

sub SetRGB # set the $colorindex in CLUT to be $newr, $newg, $newb
{
 local($Image,$colorindex,$newr,$newg,$newb)=@_;
 $Image->colorDeallocate($colorindex);
 do
 {
  $newindex = $Image->colorAllocate($newr,$newg,$newb);
 } until ($colorindex==$newindex);
}
```

Which one of the programs (a) to (e) generates output.png when given input.png as follows.

input.png: 

output.png: 

Q27)

Image before processing:

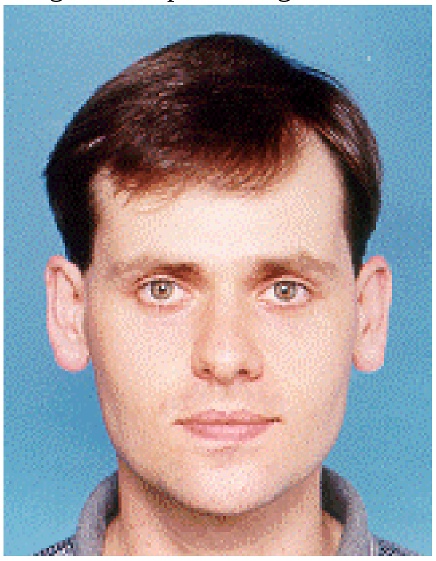
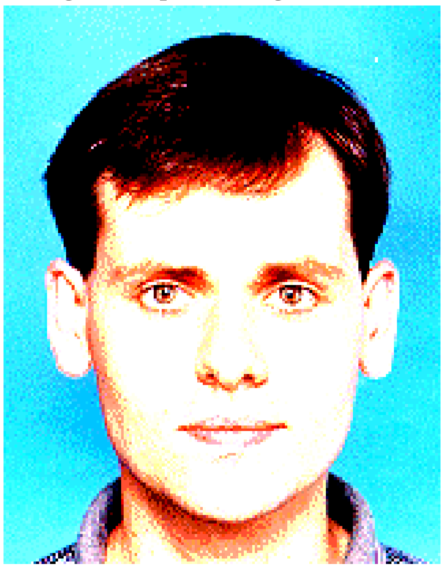


Image after processing:



What has happened?

a) The red and blue planes have been inverted, but the green plane has been unaltered
b) Image contrast has been increased by 70%
c) Hue values have been increased by 50%
d) The histogram has been equalized – "The Equalize Histogram command smoothes out the image's histogram so that it is more evenly distributed across the spectrum."
e) Saturation values have been decreased by 50%