Multimedia Programming

Medialogy, 8th Semester, Aalborg University

Wednesday 6 June 2012, 09.00 – 12.00

Instructions and notes

You have 3 hours to complete this examination.

Neither written material nor electronic equipment may be brought into the examination room.

There are 12 questions and each question is worth 10 marks. The maximum possible score is 120 marks. You must get at least 50 marks to pass.

Note that a number at the beginning of any line in a program listing or pseudocode indicates the number of that line and is not part of the program or algorithm itself.

DO NOT TURN OVER UNTIL INSTRUCTED TO DO SO!

END OF EXAMINATION

Question 1

The following pseudocode defines the insertion sort algorithm. Show that the worst-case running time of insertion sort is $\Theta(n^2)$.

```
INSERTION-SORT(A)
1  for j ← 2 to length[A]
2       do key ← A[j]
3          ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
4          i ← j − 1
5          while i > 0 and A[i] > key
6               do A[i + 1] ← A[i]
7                  i ← i − 1
8          A[i + 1] ← key
```

Question 2

Choose one of the patterns described by Gamma *et al*. (1995) and answer the following questions about it:

a. What is the name of the pattern you have chosen?

b. Give an example of a problem that your chosen pattern solves.

c. Briefly describe how the pattern solves problems like the one you mentioned in part b.

d. What are the typical costs and benefits of applying your chosen pattern?

Question 3

a. Explain, with an example, what is meant by the *race condition* in multi-threaded programming.

b. Study the following Java program and answer the questions that follow it.

```
1    public class Threads {
2      public static Integer balance = 100;
3
4      static class ChangeRunnable implements Runnable {
5          int x;
6
7          ChangeRunnable(int x) {
8              this.x = x;
9          }
10
11         public void run() {
12             try{
13                 Thread.sleep((long)Math.random()*1000);
14                 int b = balance;
15                 Thread.sleep((long)Math.random()*1000);
16                 b += x;
17                 Thread.sleep((long)Math.random()*1000);
18                 balance = b;
19             } catch(Exception e) {
20                 e.printStackTrace();
21             }
22         }
23     }
24
25     public static void main(String[] args) {
26         try {
27             Thread plus10 = new Thread(new ChangeRunnable(10));
28             Thread minus5 = new Thread(new ChangeRunnable(-5));
29             plus10.start();
30             minus5.start();
31             plus10.join();
32             minus5.join();
33             System.out.println(balance);
34         } catch (InterruptedException e) {
35             e.printStackTrace();
36         }
37     }
38 }
```

i. What are the possible outputs of this program?

ii. How would you modify the program so that it always generates the same output every time it is run?

iii. Discuss whether this modified version of the program should be considered "multi-threaded".

Question 4

a. Explain the difference between UDP and TCP. Which protocol would you use for each of the following applications:

i. A time server.

ii. FTP

iii. ping

b. Suppose we have two Java programs, Client and Server. The Server program is defined as follows:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(50000);
            Socket clientSocket = serverSocket.accept();
            PrintWriter out = new
                PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
                InputStreamReader(clientSocket.getInputStream()));
            String line = in.readLine();
            System.out.println(line);
            out.println(line+" to you too!");
            out.close();
            in.close();
            clientSocket.close();
            serverSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

(*continued*)

The Client program is defined as follows:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class Client {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost",50000);
            PrintWriter out = new
                PrintWriter(socket.getOutputStream(),true);
            BufferedReader in = new BufferedReader(new
                InputStreamReader(socket.getInputStream()));
            out.println("Hello");
            System.out.println(in.readLine());
            out.close();
            in.close();
            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

i. When running this system, which of the two programs should be started first?

ii. What is printed out to the console window by the Server program?

iii. What is printed out to the console window by the Client program?

iv. Explain the difference between a BufferedReader, an InputStreamReader and an InputStream.

v. To what does the number 50000 refer in the first line of the try block in each program?

Question 5

a. Name the three different types of thread used in a Swing program and briefly describe the function of each.

b. Study the following Java program and answer the questions that follow it.

```
1   import java.awt.BorderLayout;
2   import java.awt.event.ActionEvent;
3   import java.awt.event.ActionListener;
4   import javax.swing.JFrame;
5   import javax.swing.JLabel;
6   import javax.swing.JTextField;
7   import javax.swing.SwingUtilities;
8
9   public class GUI extends JFrame implements ActionListener {
10
11      private static final long serialVersionUID = 1L;
12      private JTextField text = new JTextField();
13      private JLabel label = new JLabel();
14
15      public GUI() {
16          setTitle("GUI");
17          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18          getContentPane().add(text);
19          getContentPane().add(label,BorderLayout.LINE_START);
20          text.addActionListener(this);
21          pack();
22          setVisible(true);
23      }
24
25      public void actionPerformed(ActionEvent event) {
26          label.setText("Text: "+text.getText());
27      }
28
29      public static void main(String[] args) {
30          SwingUtilities.invokeLater(new Runnable() {
31              public void run() {
32                  new GUI();
33              }
34          });
35      }
36  }
```

i. What kind of layout manager does the program use? Briefly describe how this type of layout manager behaves.

ii. Give the variable name and type of each component that appears in the GUI.

iii Which object in the program serves as an ActionListener?

iv. What does this ActionListener do?

v. What does the statement, “pack()”, do in the sixth line of the constructor?

Question 6

a. Let n be an integer and let $f(n)$ be a function of n. Explain precisely what it means if we say that "$f(n) = \Theta(g(n))$".

b. Show that, if $f(n) = 4n^2 + 3n$, then $f(n) = \Theta(n^2)$.

c. Give the worst case running time in terms of O-notation for the following method:

```
public static void unknown(int n) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < Math.log(n); j++)
            System.out.print(i+" ");
        System.out.println();
    }
    System.out.println();
}
```

Question 7

a. The *height* of a node in a heap is defined to be the length of the *longest simple* downward path from the node to a leaf. Why is it defined in this way, rather than just the length of the downward path from the node to a leaf?

b. State the min-heap property and draw a tree representation of an example of a min-heap with at least 7 nodes in it. How would this min-heap be stored as an array?

c. If A is an array of integers that represents a max-heap and i is the index of a node in this heap, the following pseudocode shows the MAX-HEAPIFY algorithm:

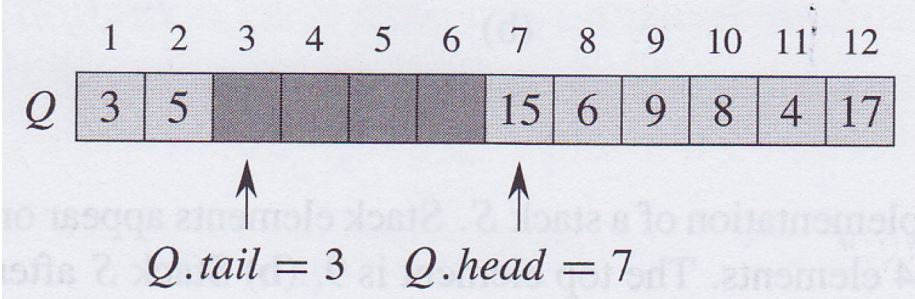
```
MAX-HEAPIFY(A, i)
 1  l = LEFT(i)
 2  r = RIGHT(i)
 3  if l ≤ A.heap-size and A[l] > A[i]
 4      largest = l
 5  else largest = i
 6  if r ≤ A.heap-size and A[r] > A[largest]
 7      largest = r
 8  if largest ≠ i
 9      exchange A[i] with A[largest]
10      MAX-HEAPIFY(A, largest)
```

Describe briefly what MAX-HEAPIFY does and explain why it has a worst-case running time of $O(\lg(n))$.

Question 8

a. Explain the difference between a stack and a queue.

b. The following array implements a queue, Q, with the head and the tail of the queue as indicated in the diagram:



i. What value will be returned by the next call to DEQUEUE on this queue? What will be the new position of the head of the queue after this call?

ii. What is the index of the position into which the next element will be inserted when ENQUEUE is next called? What will be the new position of the tail of the queue after this call?

iii. If the head position is 12 and DEQUEUE is called, what will be the new position of the head after the call?

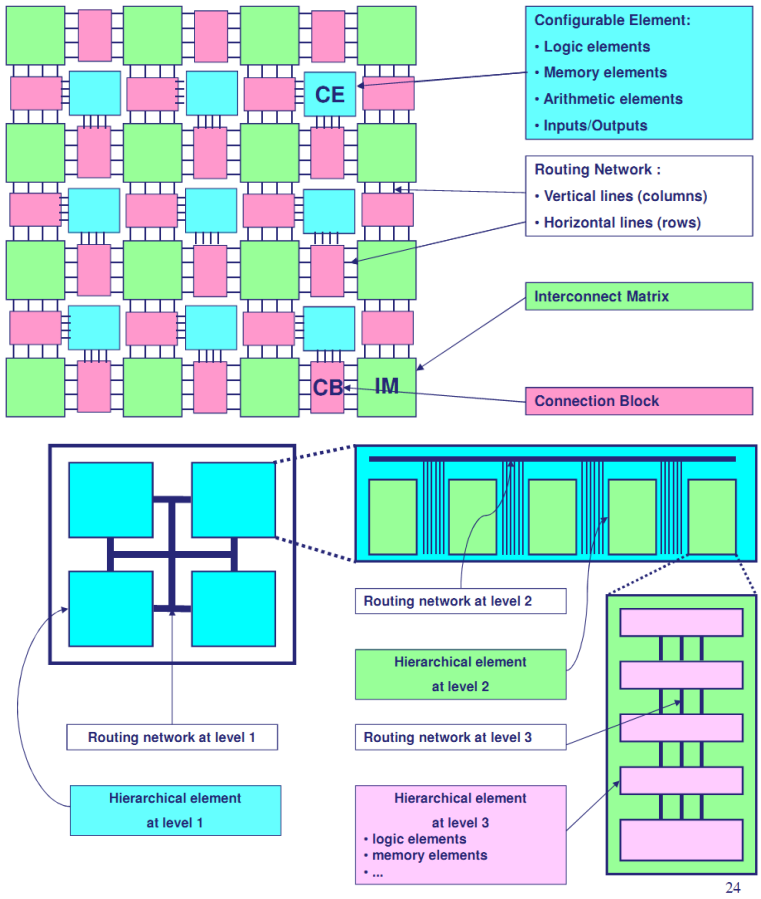
Question 9

a. FPGA stands for Field Programmable Gate Array.

i. What does "gate array" refer to?

ii. What does "field programmable" refer to?

b. Study the following figure, which shows two FPGA architecture syles, and answer the questions that follow it.

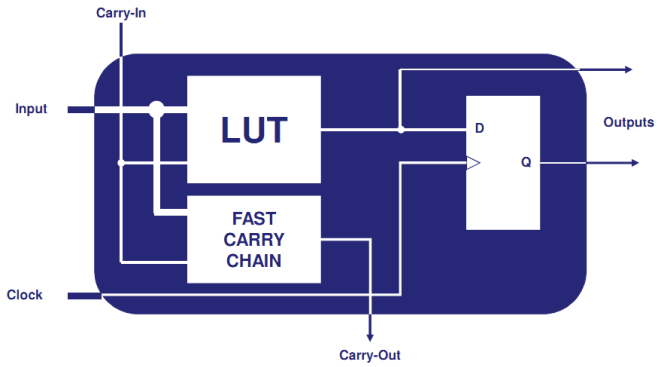


i. What are the typical architectural elements in modern FPGAs?

ii. What are their respective roles?

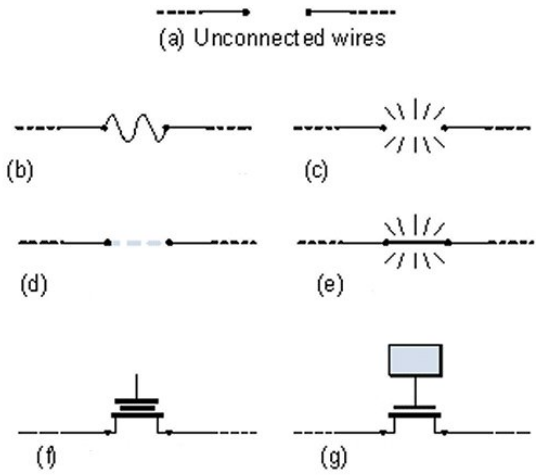
iii. What are the two architectural styles represented in the figure above?

c. The following figure shows a LUT-based configurable logic element. What is the LUT mostly used for?



Question 10

a. Study the following figure and answer the questions that follow it



Most FPGAs are based on SRAM, Flash, anti-fuse or fuse technologies.

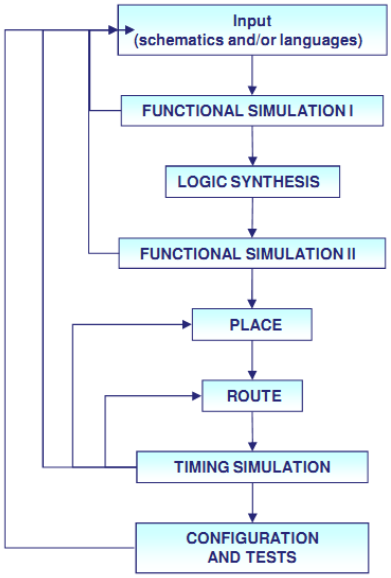
i. Identify these technologies in the figure above.

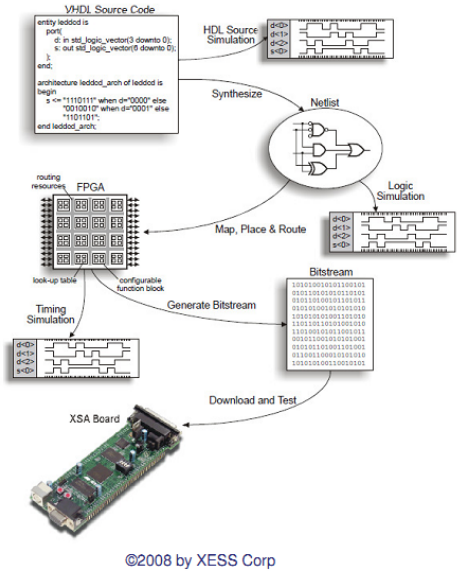
ii. What are their respective advantages and drawbacks?

iii. Which one is most commonly used?

b. What is the difference between a hard-core (hardwired) and a soft-core microprocessor?

c. Study the following figure, which shows a generic FPGA design flow, and explain what happens in each of the steps. Note that the 'map' step shown on the right hand side is not shown on the left-hand side; do not forget to describe it.





Question 11

a. What kind of problems can dynamic programming be used to solve efficiently?

b. The following algorithm is a recursive solution to the rod-cutting problem, where n is the length of the rod and p is the price array, giving the revenue for each rod length:

```
CUT-ROD(p, n)
1  if n == 0
2      return 0
3  q = -∞
4  for i = 1 to n
5      q = max(q, p[i] + CUT-ROD(p, n - i))
6  return q
```

Explain why this algorithm is inefficient.

c. Sketch an efficient dynamic programming solution to the rod-cutting problem that uses a bottom-up method. (HINT: Use an array, $r[0..n]$, to store the maximum revenues for rods of length 0 to n.) What is the worst-case running time of your solution?

Question 12

Consider this shader code for Unity:

```
(1)  Shader "Custom/NewShader" {
(2)   Properties {
(3)         _Material ("color of material", Color) = (1,0,0,1)
(4)         _Specular ("color of highlights", Color) = (1,1,1,1)
(5)   }
(6)   SubShader {
(7)         Tags { "RenderType"="Opaque" }
(8)
(9)         CGPROGRAM
(10)        #pragma surface func Lambert
(11)
(12)        half4 _Material;
(13)        half4 _Specular;
(14)
(15)        struct Input {
(16)              float dummy;
(17)        };
(18)
(19)        void func (Input IN, inout SurfaceOutput o) {
(20)              o.Albedo = _Material.rgb;
(21)              o.Specular = 0.9;
(22)        }
(23)        ENDCG
(24) }
(25) }
```

Which line(s) of the code specify the kind of this shader (vertex and fragment program in Cg, vertex and fragment program in GLSL, or Surface Shader)? On which hardware platforms (Windows PC, MacOS X computer, Android device, and/or iOS device) is this kind of shader supported?

The shader compiles without error messages. It is supposed to render a red surface with white highlights. However, no highlights are rendered. Which three(!) lines have to be edited and/or where must new lines be added to render highlights? (Specify the three line numbers.) Which parts of the three lines have to be edited and/or what should the new lines accomplish? (Instead of describing the required three changes, you can also just specify the correct code.)

Suppose the correctly edited shader computes highlights per vertex (Gouraud shading). What is the disadvantage of this computation? Propose a solution to overcome this disadvantage. (Don't provide details, just specify the general idea.) What disadvantage(s) or costs does your solution imply?

END OF EXAMINATION