COMPUTER SCIENCE TRIPOS Part II (General) DIPLOMA IN COMPUTER SCIENCE

Monday 4 June 2007 1.30 to 4.30

PAPER 10 (PAPER 1 OF DIPLOMA IN COMPUTER SCIENCE)

Answer ***five*** *questions.*

Submit the answers in five ***separate*** *bundles, each with its own cover sheet. On each cover sheet, write the numbers of* ***all*** *attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS	SPECIAL REQUIREMENTS
Script paper	*None*
Blue cover sheets	
Tags	

1 Foundations of Programming

(*a*) Distinguish between the terms *abstract class* and *interface*. [3 marks]

(*b*) Explain in detail what happens when class GPS below is instantiated noting, in particular, any assignments that are made. [17 marks]

```
class GPS
 { public int[][] a = new int[4][4];
   private int[] i = new int[1];
   private int[] j = new int[1];

   public GPS()
    { this.j[0] = this.gps(this.j, 4, new Pi(), new Fg());
    }

   private int gps(int[] i, final int N, Pass z, Feval v)
    { i[0] = 0;
      while (i[0]<N)
       { z.p(v.f());
         i[0]++; }
      return 0;
    }

   private abstract class Pass
    { public abstract void p(int n);
    }

   private class Pi extends Pass
    { public void p(int k)
       { GPS.this.i[0] = k; }
    }

   private class Pij extends Pass
    { public void p(int k)
       { GPS.this.a[GPS.this.i[0]][GPS.this.j[0]] = k; }
    }

   private abstract class Feval
    { public abstract int f();
    }

   private class Fij extends Feval
    { public int f()
       { return GPS.this.i[0]+GPS.this.j[0]; }
    }

   private class Fg extends Feval
    { public int f()
       { return GPS.this.gps(GPS.this.i, 4,
                                  new Pij(), new Fij()); }
    }
 }
```

2 Foundations of Programming

(*a*) What are the principal characteristics of a linked list? [2 marks]

(*b*) Consider a stack that consists solely of characters. A linked list can be used to model such a stack and a programmer has embarked on a class `ChLink` which begins thus:

```
class ChLink
 { private char val;
   private ChLink next;

   public ChLink()
    { this.val = '\\';
      this.next = null;
    }
```

The backstroke character '\' is reserved and marks the bottom of the stack. Why does it have to be keyed in twice? [1 mark]

(*c*) Augment class `ChLink` with two methods `push(char)` which pushes a character onto the stack and a method `pop()` which returns the character at the top of the stack and, unless it is backstroke, removes that character from the stack. It is not possible to pop the bottom marker. [10 marks]

(*d*) Write a test program which uses class `ChLink` to verify that a sequence of square brackets is properly nested. The test program might begin thus:

```
public class ChLinkEx
 { public static void main(String[] args)
    { char[] chseq = {'[','[','[','A',']',']','[','A',']'};
      ChLink start = new ChLink();
      for (int i=0; i<chseq.length; i++)
```

Opening brackets are pushed onto the stack. No other characters are pushed onto the stack but any closing bracket results in a pop and a suitable test. The test program should print either `Full Match` or `Match Error` without further explanation. Only class `ChLinkEx` need be written in this part. [7 marks]

3 Floating-Point Computation

(*a*) A hypothetical (and practically rather useless) floating-point number representation inspired by the IEEE floating-point standards uses 6 bits—one bit for sign, three bits for exponent and two (stored) bits for mantissa (significand). Assuming that 1.0 is represented in this format as 0:011:00 give the values, in decimal notation (fractions are acceptable), of all the other non-negative floating-point values in this representation. You do not need to give details of denormalised numbers or NaNs. [10 marks]

(*b*) Explain the following terms:

(*i*) absolute error;

(*ii*) relative error;

(*iii*) rounding error;

(*iv*) truncation error;

(*v*) ill-conditionedness. [5 marks]

(*c*) Assuming the floating-point representation for type `float` has b bits in its mantissa (significand), what can be said about the output of the following program?

```
float f = 10.0/3.0;
for (i=0; i<100; i++)
{ int v = (int)f;       /* get integer part of f */
  printf("%d\n", v);    /* print it */
  f = (f - v) * 10;
}
```

Discuss how accurately `f` represents 10.0/3.0 at the start of each iteration and explain which operation(s) represent the main loss of accuracy in `f` on each iteration. (You may assume that 10 significant bits of accuracy is approximately 3 decimal digits of accuracy.) [5 marks]

4 Programming in C and C++

A C programmer is working with a little-endian machine with 8 bits in a byte and 4 bytes in a word. The compiler supports unaligned access and uses 1, 2 and 4 bytes to store `char`, `short` and `int` respectively. The programmer writes the following definitions (below right) to access values in main memory (below left):

Address	Byte offset			
	0	1	2	3
0x04	10	00	00	00
0x08	61	72	62	33
0x0c	33	00	00	00
0x10	78	0c	00	00
0x14	08	00	00	00
0x18	01	00	4c	03
0x1c	18	00	00	00

```
int **i=(int **)0x04;

short **pps=(short **)0x1c;

struct i2c {
  int i;
  char *c;
}*p=(struct i2c*)0x10;
```

(*a*) Write down the values for the following C expressions:

```
**i          p->c[2]          &(*pps)[1]          ++p->i
```

[8 marks]

(*b*) Explain why the code shown below, when executed, will print the value 420.

```
#include<stdio.h>

#define init_employee(X,Y) {(X),(Y),wage_emp}
typedef struct Employee Em;
struct Employee {int hours,salary;int (*wage)(Em*);};
int wage_emp(Em *ths) {return ths->hours*ths->salary;}

#define init_manager(X,Y,Z) {(X),(Y),wage_man,(Z)}
typedef struct Manager Mn;
struct Manager {int hours,salary;int (*wage)(Mn*);int bonus;};
int wage_man(Mn *ths){return ths->hours*ths->salary+ths->bonus;}

int main(void) {
  Mn m = init_manager(40,10,20);
  Em *e= (Em *) &m;
  printf("%d\n",e->wage(e));
  return 0;
}
```

[4 marks]

(*c*) Rewrite the C code shown in part (*b*) using C++ primitives and give *four* reasons why your C++ solution is better than the C one. [8 marks]

5 Computer Graphics and Image Processing

(*a*) In image compression we use three different mechanisms to compress pixel data:

(*i*) mapping the pixel values to some other set of values;

(*ii*) quantising those values;

(*iii*) symbol encoding the resulting values.

Explain each mechanism, describe the way in which it helps us to compress the image, and describe how the mechanism is implemented in the baseline JPEG compression method. [10 marks]

(*b*) Describe the limitations of human vision in terms of:

(*i*) spatial resolution,

(*ii*) luminance,

(*iii*) colour,

and explain the implications that each of these has on the design of display devices, including numerical estimates of the limits beyond which a human cannot discriminate. [10 marks]

6 Mathematics for Computation Theory

State the requirements for $(S, \leqslant)$ to be:

(a) a *partially ordered* set;

(b) a *totally ordered* set;

(c) a *well ordered* set. [5 marks]

Let $\mathbb{N}$ be the natural numbers. Give, without proof, three examples of relations $\leqslant_i$, where $i = 1, 2, 3$, such that $(\mathbb{N}, \leqslant_i)$ satisfies exactly i of the conditions (a), (b), (c). [3 marks]

Let $S = \{a, b\}$ be an alphabet, with total order $a < b$. Let $\Sigma = S^*$ be the set of all strings over S; for $w = s_1 s_2 \ldots s_n \in \Sigma$ we write $\ell(w) = n$, and for $1 \leqslant r \leqslant n = \ell(w)$ we write $w_r = s_1 s_2 \ldots s_r$. Denote by ε the unique word of Σ such that $\ell(\varepsilon) = 0$, the *null string*. Conventionally $w_0 = \varepsilon$ for all words $w \in \Sigma$.

Define relation $\prec$ on Σ as follows:

Let $v, w \in \Sigma$, and $n = \min\{\ell(v), \ell(w)\}$. Let $r = \max\{i \mid v_i = w_i\} \leqslant n$.

Then $v \prec w$ if:

either (i) $\ell(v) = r$;
or (ii) $v_{r+1} = v_r a, \quad w_{r+1} = w_r b, \quad$ where $v_r = w_r$.

Which of conditions (a), (b), (c) above are satisfied by $(\Sigma, \prec)$? [12 marks]

7 Computation Theory

(*a*) (*i*) Define the notion of a *register machine* and the computations that it carries out. [5 marks]

(*ii*) Explain, in general terms, what is meant by a *universal* register machine. (You should make clear what scheme for coding programs as numbers you are using, but you are not required to describe a universal register machine program in detail.) [5 marks]

(*b*) (*i*) Explain what it means for a partial function f from $\mathbb{N}$ to $\mathbb{N}$ to be *computable* by a register machine. [2 marks]

(*ii*) Let $n > 1$ be a fixed natural number. Show that the partial function from $\mathbb{N}$ to $\mathbb{N}$

$$f_n(x) = \begin{cases} nx & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$$

is computable. [3 marks]

(*iii*) Explain why there are only countably many computable functions from $\mathbb{N}$ to $\mathbb{N}$. Deduce that there exists a partial function from $\mathbb{N}$ to $\mathbb{N}$ that is not computable. (Any standard results you use about countable and uncountable sets should be clearly stated, but need not be proved.) [3 marks]

(*iv*) If a partial function f from $\mathbb{N}$ to $\mathbb{N}$ is computable, how many different register machine programs are there that compute f? [2 marks]

8 Artificial Intelligence I

We have a basic search problem, consisting of a set S of states, a start state s_0, a goal test $G(s)$ that returns True if $s \in S$ is a goal and False otherwise, and a function $\exp(s)$ that returns the set of states obtained by expanding state s.

(*a*) Describe in detail the *Graph Search* algorithm for solving a problem of this type. How does it differ from the related *Tree Search* algorithm? [8 marks]

(*b*) Give a detailed description of the *Recursive Best First* search algorithm, and explain why it might be used in preference to the $A^\star$ algorithm. [12 marks]

9 Introduction to Security

(*a*) You have received a shipment of hardware random-number generators, each of which can output one 128-bit random number every 10 milliseconds. You suspect that one of these modules has been tampered with and that it actually produces only 30-bit random numbers internally, which are then converted via a pseudo-random function into the 128-bit values that it outputs.

(*i*) How does this form of tampering reduce the security of a system that uses a generated 128-bit random number as the secret key of a block cipher used to generate message authentication codes? [2 marks]

(*ii*) Suggest a test that has a more than 0.5 success probability of identifying within half an hour that a module has been tampered with in this way. [6 marks]

(*b*) Explain briefly

(*i*) the encryption and decryption steps of Cipher Feedback Mode; [3 marks]

(*ii*) why some operating systems ask the user to press a special key combination (e.g., Alt-Ctrl-Del) before each password login; [3 marks]

(*iii*) how a secure hash function can be used to implement a one-time signature scheme; [3 marks]

(*iv*) what happens if the same private key of the scheme from (*iii*) is used *multiple times*, to sign different messages. [3 marks]

10 Data Structures and Algorithms

(*a*) Give a clear description of an efficient algorithm for finding the i^{th} smallest element of an n-element vector. Write some pseudocode for the algorithm and discuss its time complexity. Compare it with other plausible ways of achieving the same result. [Notes: Use zero-based indexing. You may assume for simplicity that all the elements of the vector are different.] [4 marks]

(*b*) Give a clear description of an efficient algorithm for finding the k smallest elements of a very large n-element vector. Compare its running time with that of other plausible ways of achieving the same result, including that of applying k times your solution for part (*a*). [Note that in part (*a*) the result of the function consists of one element, whereas here it consists of k elements. As above, you may assume for simplicity that all the elements of the vector are different.] [6 marks]

(*c*) Give an optimal algorithm for solving part (*b*) for $k = 1$. Give the worst-case number of comparisons performed by your algorithm as a function of n. [Note: exact *number of comparisons*, not just asymptotic complexity.] [4 marks]

(*d*) Same as part (*c*), but for $k = 2$. [6 marks]

11 Introduction to Functional Programming

(*a*) Specify the types of the following SML functions:

(*i*) `fn f => map f` [2 marks]

(*ii*) `fn f => map map f` [2 marks]

(*iii*) `fn f => (map o map) f` [2 marks]

[Recall that the composition operator `o` has type
$(\alpha \to \beta) * (\gamma \to \alpha) \to \gamma \to \beta$.]

(*b*) Let

```
datatype α tree = empty | node of α * α tree * α tree
```

be the data type of binary trees.

(*i*) A *bijective correspondence* between types α and β is given by a pair of functions $f : \alpha \to \beta$ and $g : \beta \to \alpha$ such that g `o` $f =$ `fn` $a : \alpha$ `=>` a and f `o` $g =$ `fn` $b : \beta$ `=>` b.

Exhibit recursive SML functions `f: unit tree → (unit tree) list` and `g: (unit tree) list → unit tree` that establish a bijective correspondence between the types `unit tree` and `(unit tree) list`. [3 marks]

(*ii*) Give an alternative definition of the function `g` in terms of either `foldl` or `foldr`. [3 marks]

(*iii*) Define a `treefold` function of type

```
( α * α tree * β → β ) → β → (α tree) → β
```

and give an alternative definition of the function `f` in terms of it. [3 marks]

(*c*) Rigorously argue for the correctness of the following identities:

```
g o f = fn t:unit tree => t
f o g = fn l:(unit tree) list => l
```

[5 marks]

12 Operating System Foundations

Operating systems are integrated closely with the hardware on which they run.

(*a*) Distinguish the hardware and software involved in interrupt-driven I/O. [4 marks]

(*b*) Describe *three* uses of the interrupt mechanism in addition to device I/O. [3 marks]

(*c*) Give examples of how the privilege state bit in a CPU's status register is used. [2 marks]

(*d*) (*i*) What could go wrong in a system that does not make use of a timing device in process scheduling? [1 mark]

(*ii*) What class of scheduling algorithms would be impossible to implement without a timing device? [2 marks]

(*e*) What is the main advantage of using half the virtual address space of a process for the operating system and half for applications? [1 mark]

(*f*) Explain memory-mapped I/O. [2 marks]

(*g*) In a paging system, would you expect every page of the operating system address space to be:

(*i*) mappable in the Translation Lookaside Buffer (TLB)? [2 marks]

(*ii*) capable of being cached? [1 mark]

Explain your answers.

(*h*) How do DMA (Direct Memory Access) devices operate? [2 marks]

END OF PAPER