**CSL 101 Introduction to Computers and Programming**
Minor I, Sem I 2008-09, Max 40, Time 1 hr

**Name** __________________________ **Entry No.** __________________________ **Group**

**Note** (i) This question paper has 3 pages
(ii) Write only in the space provided below each question including back of the sheets. Use the extra sheets for rough work.
(iii) Write your answers neatly and precisely. You won't get a second chance to explain what you have written.
(iv) Note that the blank spaces do not carry equal marks. Marks will be awarded on the basis of overall correctness.

1. Given a positive integer $N$, we want to return $N^r$ which is the integer corresponding to the *reverse* of the base 10 representation of $N$. For example, if $N = 5381$, then $N^r = 1835$. Complete the following Ocaml program that reverses the input integer. **(8+2+5 marks)**
Hint: $5381 = ((5 \times 10 + 3) \times 10 + 8) \times 10 + 1$.

```
exception Negative ;;

let rec appenddig (n, rev) = if n <= 9 then (rev*10 + n)

            else appenddig (n/10 , ___rev*10 + n mod 10___)  ;;

let revint  = function n ->     (* main function *)

                 if n <= 0 then raise Negative

              else appenddig (__n , 0__) ;;
```

Partial marks 3 if initialization was correct. No partial marks for the types.
(i) Type of function `revint __int____-> __int____`

`appenddig  _int * int___-> ___int ___`

(ii) To prove the correctness of the above program, we must characterise the behavior of the function `appenddig`. Formally specify the the desired relation between the two parameters of `appenddig` that will imply the correctness of `appenddig`. (Proof is not required).
If $n = a_k a_{k-1} \ldots a_1$ then after $i$ calls to **appendig**,
`n` $= a_k a_{k-1} \ldots a_i$ and `rev` $= a_1 a_2 \ldots a_{i-1}$.
An equivalent answer is for input N, n concatenated with revint (rev) $= N^r$.
A description of the program, or a recurrence relation won't fetch any marks.

2. Write a *recursive program* `zip`$(l_1, l_2)$ that given two lists $l_1$ and $l_2$, produces a single list whose elements are pairs of corresponding elements from $l_1$ and $l_2$. For example `zip([1;2;3], [4;5;6])` returns `[(1,4); (2,5); (3,6)]`. The program should raise an exception `UnequalLengths` if the lists are of unequal length. **(2+8 marks)**
Explain on what measures(s) you will perform induction to prove the correctness and termination of your program: Induction on the *length* of the list $l_1$ and a nested induction on the length of the list $l_2$. (partial credit for indicating it is induction on the length of the lists, or just $l_1$. An induction proof was not asked for, so no credit for that.

```
exception  UnequalLengths;;
let rec zip (l1, l2) = match l1 with
```

```
(* base case *)
(* 1.1 *)          [ ] ->  if l2 = [ ] then [ ]       (*   both lists empty  *)
(* 1.2 *)                   else raise UnequalLength (* l1 empty, l2 not *)
(* induction cases *)
                   | x::xs -> (match l2 with
(* 2.1 *)                      [ ] -> raise UnequalLengths (* l2 empty, l1 not *)
(* 2.2 *)                     | (y::ys) -> (x,y) :: (zip (xs,ys))
                                     (* pair first of each list and cons it to
                                        list obtained by recurring on rest *)
                                 )
                                   (* almost all credit given for inefficient
                                      [(x,y)]@ (zip(xs,ys)).  *)
(*   No marks deducted for minor syntax errors *)
;;
```

3. Given a positive integer $N$, define an ordered partition as an **ordered** sequence $(a_1, a_2, \ldots, a_k)$ of positive numbers $\leq N$ ($0 < a_i \leq N$ ) such that $a_1 + a_2 + \cdots + a_k = N$. For example $(3, 1, 2, 1, 2)$ is an ordered partition of 9. Note that the ordered partitions $(3, 1, 2, 1, 2)$ and $(3, 2, 2, 1, 1)$ are different because the order of the numbers are different. Given a number $N$ we would like to count how may ordered partitions it has. For example, $N = 4$ has 8 ordered partitions :
$(1, 1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 3), (2, 1, 1)(2, 2), (3, 1), (4)$. **(2+5+8 marks)**

- Write all the ordered partitions of 6.
  There are 32 of them :
  (6)
  (1,1,1,1,1,1)
  (1,2,1,1,1) : 5 permutations
  (1,3,1,1) : 4 permutations
  (1,4,1) : 3 permutaions
  (1,5), (5,1)
  (2, 2, 1, 1) : 6 permuations
  (2,3,1) : 6 permuations
  (3,3)
  (4,2), (2,4)
  (2,2,2)

- Write an inductive definition for computing $P_N$, the number of ordered partitions of a positive integer $N$.

$$P_1 = 1$$
$$P_N = P_{N-1} + P_{N-2} + \cdots + P_1 + 1$$

  This implies that if $N > 1$, then $P_N = 2P_{N-1}$.
- Use this inductive definition to write a program in Ocaml which outputs the number of partitions of $N$.

```
exception Nonpositive ;;

 (*  Type of function count_partitions ____int_________ -> _____int____________   *)

let rec count_partitions  = function n ->
 if (n <= 0) then raise Nonpositive
 else if (n = 1) then 1
      else 2*count_partitions(n-1);
```

An alternate solution is to write a function directly from the recurrence (without simplifying).

```
let rec partition n =
    let rec sumpart (n , res) = if n = 0 then res else
                                sumpart (n-1 , res + partition n)
           in  if n =1 then 1 else (1 + sumpart (n-1, 0) ) ;;
```