Computer Science & Engineering Department
I. I. T. Kharagpur

Compilers: CS31003

3rd year CSE: 5th Semester (**Class Test I Answer**)

From 20:00 - 21:00 hrs *Date : 24th August, 2011*

Roll No.: **Name:**

Marks out of 20:

1. Write a recursive OCAML function `printL l n` that takes two parameters `l`, an integer list and `n` a positive integer. It prints the first n integers of the list. [**3**]

```
val printL : int list -> int -> unit = <fun>
```

```
let rec printL l n =
    if l = [] or n = 0 then Printf.printf "\n"
    else begin
        Printf.printf "%d " (List.hd l);
        printL (List.tl l) (n-1)
    end
;;
```

2. Following are two incomplete OCAML functions. When `main()` is i called, it reads n integers in a list. It also reads an integer r, $0 \leq r \leq n$. It calls the function `ncrEnum()` to enumerates and print, using `printL()`, the $\binom{n}{r}$ combinations of the set of n integers. Complete the function by writing appropriate expressions in place of $\alpha_1, \cdots, \alpha_7$. [**7**]

```
let rec ncrEnum data comb combI n r =
    if r = 0 then begin printL comb combI; 1 end
    else if r = n then begin α1; 1 end
         else begin
             let dataTail = α2  in
             (ncrEnum dataTail α3 (combI+1) (n-1) (r-1)) +
             (ncrEnum dataTail α4 α5 (n-1) r)
         end
;;
let main() =
    print_string("Enter a +ve integer: ");
    let n = read_int() and l = ref [] in
    begin
        Printf.printf "Enter %d integers\n" n;
        for i = 1 to n do  l := α6 done;
        Printf.printf "Enter r, 0 <= r <= %d\n" n;
        let r = read_int() and comb = [] in
        begin
            Printf.printf "nCr: \n";
            Printf.printf "count: %d\n" α7
        end
    end
;;
```

The input-output behaviour of the program is as follows:

```
# main();;
Enter a +ve integer: 4
Enter 4 integers
1
2
3
4
Enter r, 0 <= r <= 4
3
nCr:
2 3 4
1 3 4
1 2 4
1 2 3
count: 4
- : unit = ()
```

α_1	`printL (comb @ data) (combI+n)`
α_2	`List.tl data`
α_3	`(comb @ [List.hd data])`
α_4	`comb`
α_5	`combI`
α_6	`!l @ [read_int()]`
α_7	`(ncrEnum !l comb 0 n r)`

3. Write a *Python* function `printL(data,n)` that will print the first n elements of the list `data`. **[3]**

```
def printL(data, n):
    x = ''
    for i in range(n): x = x + data[i] + ' '
    print x
```

4. The following incomplete *Python* program reads n integers(in a list) and an integer r, $0 \leq r \leq n$. It enumerates and prints (using `printL()`) the $\binom{n}{r}$ combinations of the set of n integers. Complete the program by writing appropriate expressions and statements in place of $\beta_1, \cdots, \beta_7$. **[7]**

```
#! /usr/bin/python
# Enumerating nCr
# definition of printL() comes here
def ncrEnum(data, comb, combI, n, r):
    if r == 0:
       printL(comb,  β1)
       return 1
    if r == n:
       for i in range(n): β2
       printL(comb, combI+n)
       return 1
    comb[combI] = β3
    count = ncrEnum(β4, comb, combI+1, n-1, r-1)
    count = count + β5
    return count
def main():
    print 'Enter integers separated by blanks: '
    x = raw_input()
    data = β6   # create list of integers
    n = len(data)
    comb = [i for i in range(n)]  #creates array of size n
    print 'Enter a +ve integer r, 0 <= r <=', n
    r = input()
    print 'nCr: '
    nCr = β7
    print 'count: ', nCr
main()
```

The input-output behaviour is as follows:

```
Enter integers separated by blanks:
1 2 3 4
Enter a +ve integer r, 0 <= r <= 4
2
nCr:
1 2
1 3
1 4
2 3
2 4
3 4
count:  6
```

β_1	`combI`
β_2	`comb[combI+i] = data[i]`
β_3	`data[0]`
β_4	`data[1:]`
β_5	`ncrEnum(data[1:], comb, combI, n-1, r)`
β_6	`x.split()`
β_7	`ncrEnum(data, comb, 0, n, r)`

Rough Work