Indian Institute of Technology, Delhi

FALL, 2016

COL 100: INTRODUCTION TO PROGRAMMING

Minor 1

One Hour

NOTE:

- All answers need to be brief and to the point.
- Please make any assumptions that you deem to be reasonable.
- Follow the *spirit of the question*. Do not immerse yourself in irrelevant details.
- Every answer needs to be written neatly and cleanly in the space provided for it.
- If required, please finalize the code in the space provided for rough work, and then write it cleanly in the space provided for writing your C statements.
- Use proper handwriting, and do not write anything on the margins.
- Note that in some questions, we might not have part marking.
- You are not allowed to carry any electronic gadgets including calculators and mobile phones.
- The closer your answer is to the model answer in terms of the lines of code, the more marks you get.
- The final answer needs to be **written with a pen**.
- There are three additional pages at the end for rough work.
- Even if your final answer is correct, you might not get full marks (or any marks) if the explanation is wrong.
- This question paper **NEEDS TO BE SUBMITTED**. Do not take it with you.

Total Marks: 30
Total Number of Pages : 10

Name:	Group No:	Entry No:

Marks:	1	2	3	4	5	6	Total

1. Consider two variables: a and b. They can take integer values between 1 to 4. When will the following conditions evaluate to true? Give your answer in terms of pairs of values (a,b). Example: The condition ($a > b$) will evaluate to true with the pairs: (4,3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1) (5 marks)

NO PART MARKING

(a) ($a < b$) Ans: (1,4) (1,3) (1,2) (2,4) (2,3) (3,4)

(b) ($a < b$) && ($b > a$) Ans: (1,4) (1,3) (1,2) (2,4) (2,3) (3,4)

(c) ($a < b$) || ($b > a$) Ans: (1,4) (1,3) (1,2) (2,4) (2,3) (3,4)

(d) ($a - b == 2$) Ans: (4,2) (3,1)

(e) ($a - b == 2$) && ($a > 2$) Ans: (4,2) (3,1)

2. Consider the grid below, where the x and y axes can have values from $1 \ldots 8$. Write a condition, which will evaluate to true for x and y values marked as T, and evaluate to false otherwise. This is a bad solution and not acceptable: ($(x == 1$ && $y == 1)$ || $(x == 2$ && $y == 1)$ || $(x == 3$ && $y == 1) \ldots$). Try to minimize the number of terms in the expression. Let the x axis be horizontal and let the y axis be vertical. The answer is not complete without a proper explanation. Even if the answer is right, you might not get full marks unless the explanation is correct, easy to understand, and mathematically sound. You can assume that the computer knows that $1 \leq x, y \leq 8$. The upper left corner is (1,1), and the bottom right is (8,8). (5 marks)

	1	2	3	4	5	6	7	8
1	T	T	T					
2	T	T	T				T	T
3	T	T	T			T		T
4					T			T
5				T				T
6			T					T
7		T						T
8		T	T	T	T	T	T	T

Answer: $((x \leq 3)\&\&(y \leq 3))$ || $(((x + y == 9)$ || $(x == 8)$ || $(y == 8))\&\&(x * y \neq 8))$

Explanation of the Solution

- The expression for the square on the upper left: $(x \leq 3)\&\&(y \leq 3)$
- Expression for the diagonal: $x + y == 9$
- Expression for last row or last column: $(x == 8)$ || $(y == 8)$
- Expression for a cell in the diagonal or the last row or the column is: $(x + y == 9)$ || $(x == 8)$ || $(y == 8)$.
- Exclusion: corner squares (1,8) and (8,1). The following expression works: $x * y \neq 8$ This also excludes (2,4) and (4,2). That is alright. They any way do not have a T.
- The expression for the squares that contain a T in the diagonal or the last row or the last column is: $((x + y == 9)$ || $(x == 8)$ || $(y == 8))\&\&(x * y \neq 8)$.
- Combining all the conditions we have: $((x \leq 3)\&\&(y \leq 3))$ || $(((x + y == 9)$ || $(x == 8)$ || $(y == 8))\&\&(x * y \neq 8))$

3. Write a program, which takes a positive integer as input, and prints which powers of 2 does the number lie between. For example, the number $\boxed{269}$ lies between $2^8(256)$ and $2^9(512)$. If the input is 269, the output should be $\boxed{8\ 9}$. Borderline cases which are powers of 2, such as $\boxed{256}$, should be aligned to the lower limit of the desired range output, ie. $2^8(256)$ and $2^9(512)$. Just write the contents of the main function. You should not use math functions such as pow. Hint: Use a loop

How would you modify the program to take the base as an input as well, and print which powers of the base does it lie between? For example, with a base of 10, 40,000 lies between 10^4 and 10^5. You do not have to write the program again, just indicate what simple changes need to be made to your program. (4 marks)

Answer:

```
void main(){
/* declare variables*/
    int exp = 0, pow2 = 1;
    int num;

/* read the number */
    scanf("%d",&num);

/* loop */
    while (1){
/* check if the number is already between pow2 and 2*pow2 */
        if( (num >= pow2) && (num < 2*pow2)){ /* 1 mark */
            printf ("%d %d\n",exp, exp+1); /* 1 mark */
            exit(1);
        }

/* otherwise start from the next power of 2*/
        exp++; /* 0.5 mark */
        pow2 = pow2*2; /* 0.5 mark */
    }
}
```

With an arbitrary base. We will just replace all the expressions of the form (2 * pow2) with (base * pow2). (1 mark)

Note to TAs: Deduct marks if there are any mistakes in the code. Make a fair and consistent scheme.

4. Answer the following questions about the program below. Note that some of these question (with an answer box) do not have any part marking. (6 marks)

1 mark per sub-part

```
void main() {
        int n;

        printf("Enter the number: ");
        scanf("%d", &n);

        while (n != 0) {
                printf("%d ", n % 2);
                n = n / 2;
        }
}
```

(a) Does the program print the binary representation of a positive number in reverse order: (Y/N)? Ans: Y

(b) If the answer is Yes, then why?

Consider a binary number of the form: $n = x_k 2^k + \ldots + x_1 2^1 + x_0 2^0$. In the first iteration, we compute: $n\%2 = x_0$ (rest of the terms are divisible by 2). We thus get the first binary digit. Now, let us use mathematical induction and assume that just after the while statement (in the $(i+1)^{th}$ iteration), we have $n = x_k 2^{k-i} + \ldots + x_i 2^0$. Subsequently, we compute $n\%2$, which is equal to x_i (again because 2 divides the rest of the terms). Then, we divide n by 2. Thus in the $(i+2)^{th}$ iteration just after the while statement: $n = x_k 2^{k-i-1} + \ldots + x_{i+1} 2^0$. This is because $(x_i 2^0)/2 = x_i/2$, which is equal to 0 ($x_i \in [0, 1]$). We thus observe that the induction hypothesis holds, and in every iteration i we print x_{i-1}. These are precisely all the binary digits in the binary representation of the number, n.

(c) What is the output, if the input is 37? Ans: 1 0 1 0 0 1

(d) How would you modify the program to print the 1's complement of the number? Make as few modifications as possible. You can use the ! operator if you wish. It converts a 0 to a 1, and a 1 to a 0.

Replace: printf("%d ", n%2) with printf("%d ", !(n%2))
Note to TAs: The order of bits does not matter (forward or reverse). If a student has done something to print them in the forward order, ignore that piece of the code.

(e) How can we print the octal representation of a positive integer? Show the modified code.

Make these changes to the code:

```
printf("%d ", n % 8);
n = n / 8;
```

Note to TAs: Again the order of digits does not matter (forward or reverse). If a student has done something to print them in the forward order, ignore that piece of the code.

(f) How many binary digits will be printed for the maximum positive value that the number n can take on standard compilers where the size of an int data type is 4 bytes? Be very accurate. Ans: 31

5. [**tricky**] Consider a 13-bit number system. It has the digits 0-9, and the following extra digits, A (10), B(11), and C(12). Perform the following subtraction, and write the answer below the line. Formally, justify and explain your steps. Note that converting the numbers to decimal, doing the subtraction, and converting the numbers back, is clearly not an option. [HINT: Think about how subtraction (with borrows) is done in base-10. Apply a similar kind of thinking here.] (5 marks)

	A	8	0	0	3
—		B	3	9	6
	9	9	9	3	A

Explanation: Consider the first two digits – 3 and 6. 3 is less than 6. We thus need to borrow. The nearest position that we can borrow from is column 4. We use the result (in base 13): 8003 = 7003 + 1000 = 7003 + 0CC0 + 10 = 7CC3 + 10. We thus have: $A8003 = A7CC3 + 10$.

Let us thus add 10 (13 in base 10) to the least significant digit, 3. The result is 13+3 = 16 in base 10. Now, we subtract 6 from 16 to get 10. 10 is A in base 13. Thus, the least significant digit of the result is A.

Let us now consider the second column of the number $A7CC3$. We need to subtract 9 from C. We get 3.

Let us now consider the third column of $A7CC3$. We need to subtract 3 from C. We get 9.

Let us now consider the fourth and fifth columns. We need to compute: A7 - B. $A7 - B = 90 + 17 - B$. 17 in base 13 is 20 in base 10. B is 11 in base 10. Thus, $A7 - B = 9$. Thus, the fourth digit in the result is 9. Since we add 90 to this number, the fifth digit is 9.

6. [**difficult**] Consider an n bit number system that uses the 2's complement representation. We can represent any number(A) as: $a_n a_{n-1} \ldots a_2 a_1$, where a_n is the n^{th} (most significant) digit and a_1 is the least significant digit. For example, in a 4 bit number system, we can have a number of the form: 1100. Here, $a_4 = 1, a_3 = 1, a_2 = 0$ and $a_1 = 0$. Prove that $A = -1 \times a_n \times 2^{n-1} + \sum_{i=1}^{n-1} a_i 2^{i-1}$. Consider two examples in a 4-bit number system: 7 (0111) = 0 + 4 + 2 + 1, -3 (1101) = 1 *(-8) + 4 + 1. (5 marks)

Answer:

Case I: Consider a number u($u < 0$). Let the unsigned form of its n bit 2's complement representation be the number A. (If $n = 4$, $u = -3$, then the representation of u is 13 ($\therefore A = 13$)).

$$\begin{aligned} A &= a_n a_{n-1} \ldots a_2 a_1 \\ &= 2^{n-1} + \underbrace{a_{n-1} \ldots a_2 a_1}_{A'} \quad (u < 0 \Rightarrow (a_n = 1)) \\ &= 2^{n-1} + A' \end{aligned}$$

We need to prove that: $\boxed{u = -2^{n-1} + A'}$. This is because A' is equal to the binary representation of the first $n-1$ digits. It is thus equal to: $\sum_{i=1}^{n-1} a_i 2^{i-1}$.

Let us consider the unsigned form of the 2's complement representation of u, which is A.

$$\begin{aligned} A &= 2^n - \mid u \mid \quad (by\ definition) \\ \Rightarrow A &= 2^n + u \quad (u = -\mid u \mid, and\ u < 0) \\ \Rightarrow u &= -2^n + A = -2^n + 2^{n-1} + A' = -2^{n-1} + A' \end{aligned}$$

Hence, proved.

CaseI detailed:

Let us consider the unsigned form of the 2's complement representation of u, which is A.

$$A = 2^{n-1} + A'$$

Also, the 2's complement representation of a number is calculated by inverting the bits of the modulus of the number, and adding one.

$$\begin{aligned} Let \ \mid u \mid &= b_n b_{n-1} \ldots b_2 b_1 \\ \mid u \mid &= 2^{n-1} * b_n + 2^{n-2} * b_{n-1} + \ldots + 2^1 b_2 + 2^0 b_1 \end{aligned}$$

Thus the unsigned form of its 2's complement representation, i.e. A would be

$$\begin{aligned} A &= 2^{n-1} * (1 - b_n) + 2^{n-2} * (1 - b_{n-1}) + \ldots + 2^1 (1 - b_2) + 2^0 (1 - b_1) + 1 \\ A &= (2^{n-1} + 2^{n-2} + \ldots + 2^1 + 2^0) - (2^{n-1} * b_n + 2^{n-2} * b_{n-1} + \ldots + 2^1 b_2 + 2^0 b_1) + 1 \\ A &= 2^n - 1 + 1 - \mid u \mid = 2^n + u \quad (since\ u < 0) \\ \Rightarrow u &= -2^n + A = -2^n + 2^{n-1} + A' \quad (proved\ earlier) \\ &= -2^{n-1} + A' \end{aligned}$$

Hence, proved.

CaseII: ($u \geq 0$)
In this case the MSB (a_n) is 0. We thus need to prove that: $u = \sum_{i=1}^{n-1} a_i 2^{i-1}$. This follows from the definition of the 2's complement number system. The MSB is 0 for all non-negative numbers, and the unsigned representation of a non-negative number u is the number u itself. A maximum of $n-1$ bits are allotted for the representation of such numbers. The value of this $n-1$ bit binary number is: $\sum_{i=1}^{n-1} a_i 2^{i-1}$, which is the same as u. Thus, this case is trivially true.
