Indian Institute of Technology Mandi
IC150: Computation for Engineers
Tutorial 1 Model Solutions

Note: For exercises 11-15 that ask you to design and write a program, work systematically: first work out some examples, next identify the issues to be tackled, then devise the algorithm and identify the necessary variables. Finally, write the code and hand simulate to ensure that it is correct.

1. Fill in the blanks:

   a) The auto-increment operators are ___`++var` and `var++`______________.

   b) The escape sequence `'\t'` represents __tab____________.

   c) The ______%________ operator cannot be applied to a `float` or `double`.

2. For each of the following, state whether it is a valid C variable name. If it is not valid, explain why.

   a) 6thBTech Invalid, variable name shouldn't start with number. It should start with alphabet.

   b) BoysNgirls Valid

   c) ladies&gents Invalid, variable name shouldn't contain any special character except underscore (_).

   d) Num_Hostel-Rooms Invalid, variable name contains special character (-).

3. Choose the right answer(s). The **break** statement is used to exit from:

   a) an **if** statement

   b) a **for** loop

   c) a program

   d) the **main()** function

4. For each of the following statements, indicate True or False

(1) Each new C instruction has to be written on a separate line	False
(2) Usually all C statements are entered in lower case letters	True
(3) Blank spaces may be inserted between two words in a C statement	True
(4) Blank spaces cannot be inserted within a variable name	True

5. Do the indicated conversions of C constants:

   (b) 10100101b to decimal **Ans:** 165

   (d) 395 to binary **Ans:** 10001011

6. Write the equivalent using `while`:

```
for(i=0; i<10; i++)
 {
 // code to do something
 }
```

```
i=0;
while(i<10)
{
     // code to do something
     i++;
}
```

7. What do these loops print?

```
for(i = 0; i < 10; i = i + 2)
        printf("%d\n", i);
```

Ans:

```
0
2
4
6
8
```

```
for(i = 100; i >= 0; i = i - 10)
        printf("%d\n", i);
```

Ans:

```
100
90
80
70
60
50
40
30
20
10
0
```

8. Hand-simulate the following code showing the memory contents after execution of the **underlined** code. What is the output of the program?

```
int main()
  {
    int i = 0, j = 0;
    do {
        i++;
        if (i%3 == 2) j = i + j;
    } while (j < 10);
    printf("%d %d\n", i, j);
  }
```

Ans:

i	**j**
1	0
2	2
3	2
4	2
5	7
6	7
7	7
8	15

Output: 8 15

9. If `a = 10, b = 12, c = 0,` find the values of the following expressions:

```
(1) a != 6 && b > 5       1
(2) a == 9 || b < 3       0
(3) !( a < 10 )           1
(4) !(a > 5 && c)         1
(5) 5 && c != 8 || !c     1
```

10. What is the output of:

```
int main( )
{
 int x=4,y,z;
 y = --x;
 z = x--;
 printf("%d%d%d\n", x, y, z);
}
```

Ans:

```
2
3
3
```

11. Write a program to print all the ASCII values and their equivalent characters using a while loop. The ASCII values vary from 0 to 255.

```
#include <stdio.h>
int main()
{
        int i;
        for (i = 0; i < 256; i++)
                printf("%d %c \n",i,i);
}
```

12. One technique for encrypting text is *rot13*. Each alphabetic character is replaced by the character 13 positions ahead in the alphabet. “Ahead” is done with wraparound, i.e. 'a' is considered to be 1 position ahead of 'z'. E.g. given “abc” the rot13 output is “nop” and “time” yields “gvzr”. Note that by applying rot13 encryption twice we get back the original. Write a program that reads one line of text from the terminal and prints the rot13 encrypted output. Assume that only lowercase alphabets are entered.

```
#include<stdio.h>

int main()
{
     char ch, ench; // Original char, Encrypted char

     printf("Enter the text for encryption: ");
     ch = getchar();

     while(ch!='\n')
     {
          if(ch<='m')               // 13 positions ahead
               ench = ch+13;
          else      // 13 positions ahead with wraparound
               ench = ch+13-26;

          putchar(ench);
          ch = getchar();
     }
     putchar('\n');
}
```

13. Design a program that reads non-negative integers one at a time from the terminal. As it reads each integer, it keeps track of the average and the maximum. The end of the list of numbers is indicated by entering a negative integer (which is not counted). After this, the program prints out the average and the maximum. Your program should work for any number of input integers.

```
#include<stdio.h>
int main()
{
     int iNum,count=0,max=0; // count the integers, max:maximum
     float avg=0,sum=0; // avg:average, sum: sum of integers
     printf("Enter an integer iNumber: ");
     scanf("%d",&iNum);
     while(iNum>=0)
     {
          count++; // Increment the counter
          sum=sum+iNum; // Add the number to calculate the
average
          if(max<iNum)
               max=iNum; // Store the maximum number
          printf("Enter an integer iNumber: ");
          scanf("%d",&iNum);
     }
     if(count>0)
          avg=sum/count; // Calculate the average
     printf("Average=%g, Maximum=%d\n",avg,max);
}
```

14. The cutoffs used in grading a particular course are: below 40: F; 40 to 44: E; 45 to 54: D; 55 to 64: C ; 65 to 74: B; 75 to 84: A; 85 and above: O.
    Given the variable `marks`, write C code using only `if ... else` to assign `'O'`, `'A'`, `'B'`, `'C'`, `'D'` `'E'` or `'F'` to the variable `grade`.

```
     // Set the cutoff marks and corresponding grades
     if (marks<40)
          grade='F';
     else if (marks>=40 && marks<45)
          grade='E';
     else if (marks>=45 && marks<55)
          grade='D';
     else if (marks>=55 && marks<65)
          grade='C';
     else if (marks>=65 && marks<75)
          grade='B';
     else if (marks>=75 && marks<85)
          grade='O';
     else
          grade='A';
}
```

15. Given the sides of a rectangle, write code for the function `IsAreaBigger()` that returns true if the area of the rectangle is bigger than its perimeter, false otherwise.

    ```
    int IsAreaBigger(int len, wid)
    ```

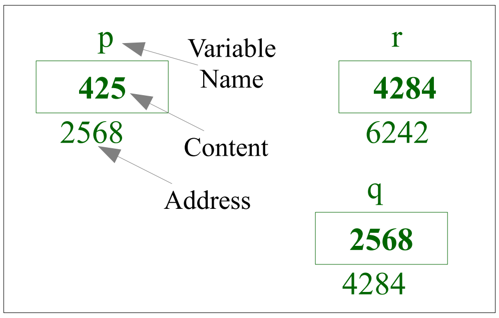
```
{


}
```

```
int IsAreaBigger(int len, int wid)
{
     if((len*wid) > 2*(len+wid))
          return 1;
     else
          return 0;
}
```

16. Given the information in the table on the left below, draw a picture of memory showing the addresses, contents and variable names. Next, for each expression in the table on the right, write its value.

Name	Address	Contents
p	2568	425
q	4284	2568
r	6242	4284

Expression	Value
**r	425
&p	2568
&(*r)	4284
*(&q)	2568


p
Variable Name
425
2568
Content
Address
r
4284
6242
q
2568
4284