BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)

First Semester 2017-18

10-Oct-2017
Closed-book type

CS F111 Computer Programming
Mid-Semester Test

45 marks (22.5%)
4:00 – 5:30 PM

1. A bit pattern stored in computer memory is interpreted differently depending on the context. Consider the 32 bits abbreviated by the hexadecimal notation 0xC0000000.

a. If this 32-bit entity represented a signed integer (in 2's complement form), what value would it represent? Show all steps how you arrived at the answer. (Note: you may write the final answers in powers of 2.) **[3]**

b. If these 32 bits represented a floating point number (that used the IEEE-754 format), what value is stored at the memory location? Show all steps. **[3]**

2. You have learned about the Goldbach conjecture in class. Now, you will learn about another unsolved problem in mathematics – the Collatz conjecture.
Let n_0 be any positive integer. For i = 0, 1, 2, … define $n_{i+1} = n_i/2$ if n_i is even; $3n_i + 1$, if n_i is odd. The conjecture is that no matter what the value of n_0 is, the sequence will always reach 1. Numbers that are generated this way are called "hailstones". For instance, starting with n_0=12, one gets the sequence 12, 6, 3, 10, 5, 16, 8, 4, 2, 1 – a total of 10 hailstones. Starting with n_0=19, for example, takes longer to reach 1: 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 – a total of 21 hailstones. Write a C program that take a positive integer from the user, and generates and prints the hailstone sequence for that integer. The program should then prompt for the next integer, and continue till the user inputs a zero or a negative value.

You should write and use a function of the following prototype in your program:

```
int generateHailstones(int n); /* prints the hailstone sequence
                                  beginning with n, and returns the numbers
                                  of terms in the sequence */
```

[6]

3. The GDP growth rate of a country (expressed in %) – considered the most important indicator of the country's economic health, is measured every quarter, but is averaged annually. Given the average annual GDP growth rates of India for several years (earliest year to latest), you have to write a C program to do the following tasks:

a. To identify the longest sequence of increasing GDP rates, and print out all the individual GDP growth rates in that sequence, along with the year each rate corresponds to. If there are two or more sequences that are of equal length, your program should print the most recent one. **[5]**

b. To find out the two years (from the given data) between which the GDP growth rate had the steepest peak or dip (i.e., either an increase or fall), and print the individual GDP rates of both years, as well as the difference (absolute value). **[3]**

The program should ask the user how many years of annual GDP growth rates is to be taken, (`num` <=100), and take the years and corresponding GDP rates in two arrays. **[2]**

Sample output (array input is not shown):

```
The most recent longest sequence of successively increasing average
annual GDP rates:
2002 : 3.803975%
2003 : 7.860381%
2004 : 7.922943%
2005 : 9.284825%
```

```
Largest difference in GDP growth rates was between 1979 and 1980.
1979 : -5.238183%
1980 : 6.735822%
Difference in GDP rates : 11.974004%.
```

4. The function `bitcount`, whose implementation is incompletely shown below, counts the number of 1-bits in its integer argument and returns that count:

```
int bitcount(unsigned x)
{
  int b;
  for (b = 0; ___(i)___; x >>= 1)
        if (x & 01)
              ___(ii)___;
  return b;
}
```

a. Fill in the blanks indicated by (i) and (ii) with an appropriate expression each. **[2]**

b. What is the significance of receiving the argument as an unsigned integer? **[1]**

5. *Predict the output of each of the following code segments and justify your answer briefly. If an error, indicate what type and why. No justification, no marks will be awarded.* **[9]**

(a)

```
int main()
{
 int a = 1, b = 2, c = 3,
  d = 4, e;
 e = c + d = b * a;
printf("%d, %d\n", e, d);
}
```

(b)

```
 int main()
{
  int i ,j, *p1, *p2;
  p1 = &j;    p2 = &i;
  *p1 = 14;  *p2 =10;
  *p2=*p2 + *p1 * *p2;
  printf("%d", i+j);
}
```

(c)

```
int main()
{
int a =5, b=0;
b = (b , a++) || (a = 0);
printf("%d %d\n", a, b);
}
```

(d)

```
int i=10,*p=&i;
int main()
{
  foo(i);
  printf("%d\n",i);
  printf("%d\n",*p);
}
void foo (int i)
{  i = ('A' ? 1 : 0);
  (*p)++;
}
```

(e)

```
int main()
{
    int a =15, b =10;
    int c = 5;
    if(a > b > c)
      printf("True");
    else
     printf("False");
}
```

(f)

```
int main()
{
unsigned int a = 60;
unsigned int b = 13;
printf("\n %d  %d", a&b,
a|b);
}
```

6. In order to obtain the pattern given on the right panel, the following code snippet was used (N has been `#defined` as 7):

```
for (i = 0; i < N; ++i)
{
   for (j = 0; j < N; ++j)
          ___(a)___? printf("* ") : printf("  ");
   ___(b)___;
}
```

```
*           *
* *       * *
*   *   *   *
*     *     *
*   *   *   *
* *       * *
*           *
```

Complete the code by filling an expression in (a) and a function call in (b). Simply write down (a) and (b) in your answer book. **[2 + 1]**

7. *Answer the following questions briefly:*

a. Write a Linux shell command to delete all files except those having a **`.c`** extension from the current directory. **[1]**

b. Write a Linux command to display all lines in **`file1.txt`** that begins with any letter of the English alphabet and ends with a non-numeric character. **[1]**

c. Suppose your home directory is **`/home`** on your Linux system, and you are currently located there. Assuming you have all necessary permissions, what will be the resultant Unix tree structure after you sequentially execute the following commands at the shell prompt? **[2]**

```
mkdir bin
touch ./bin/f1.c
mkdir tmp
cat > /home/tmp/f2.c [Assume a line of text is entered]
mkdir /bin
cd /bin
touch f1.h
cd ~
mkdir ./../usr
cd ./../usr
mkdir user1
```

d. When attempting to print the average of `n` integers whose sum is stored in the integer variable `sum`, why does the statement `avg = sum / n;` result in an imprecise answer, despite declaring `avg` as a float and using the "`%f`" format specifier in `printf()`? What should be done to rectify the situation? **[1]**

e. Consider `pch`, `pshort` and `pdouble` declared as pointers to `char`, `short int` and `double`, respectively. Arrange the three variables in increasing order of size. Explain your answer. **[1]**

f. Seemingly compact, cryptic (and possibly even fancy!) statements such as the following are best avoided by experienced programmers. Why so? **[2]**

```
a[i] = i++;
c = a++ + ++a + a;
```

For bonus credit! **[2]**

At line 7, the correct value stored at `ch` is printed, whereas at line 8, the output is not 10. Can you explain this?

```
int main()
{
   char ch=10, *pchar=&ch;
   short si, *pshort;
   si = ch;
   pshort = pchar;
   printf("%d\n", si);      /* line 7 */
   printf("%d\n", *pshort); /* line 8 */
}
```