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

First Semester 2017-18

10-Oct-2017
Closed-book type

CS F111 Computer Programming
Mid-Semester Test Answers

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. The binary expansion of OxC0000000 is 1100 0000 0000 0000 0000 0000 0000 0000. Because the MSB is 1, it represents a negative integer in 2's complement. The magnitude of this number is (0011 1111 1111 1111 1111 1111 1111 1111) + 1 = 0100 0000 0000 0000 0000 0000 0000 0000 = 2^{30}. So, the value is **-2^{30}**. **[3]**

b. To interpret the representation as IEEE-754 floating-point number:

1 10000000 0… (23 zeros)

Biased exponent = 2^7 = 128; exponent = 128 – 127 = 1

Since the sign bit is 1, the number is negative. So, the floating-point number is: –1.0 x 2^1 = **-2.0** **[3]**

2. *C program for testing the Collatz conjecture:* **[6]**

```
#include <stdio.h>
int generateHailstones(int seed)
{
int count = 1, term = seed;
printf("%d", term);
while (term != 1) /* keep generating terms till it reaches 1 */
        {
        if (term % 2 == 0)  /* even number */
                term = term / 2;
        else
                term = 3 * term + 1;
        printf(", %d", term);
        count++;
        };
putchar('\n');
return count;
}

int main()
{
int val, count;
do {
        printf("Enter a +ve integer to generate the hailstone sequence: ");
        scanf("%d",&val);
        if (val <= 0)
        {
         printf("Hailstone sequences only for +ve numbers. Bye!\n\n");
         break;
        }
        count=generateHailstones(val);
        printf("The number of terms = %d\n\n",count);
    } while (1);
}
```

3. *GDP rates problem:*

```
#include <stdio.h>
#define MAX 100
int main()
{
int i, num, j, count, max_count = 0, max_index, year[MAX], max_diff_index;
double arr[MAX], diff, max_diff;
char ch;
scanf("%d",&num);
/* Taking inputs into arrays */                                              [2]
for (i=0; i<num; ++i)
{
  scanf("%d",&year[i]);           /* taking array input for year */
  scanf("%lf",&arr[i]);           /* taking array input for GDP rate */
}
/* Part (a) of the question */                                               [5]
for (i=0; i<num; ++i) /* for each GDP rate */
{
  for (j=i, count=0; j < num-1; ++j)  /* examine all successive rates… */
    if (arr[j] <= arr[j+1])       /* non-descending values so far */
        count++;      /* keep track of how many elements in the sequence */
    else
         break;       /* found a lower rate, time to stop the sequence */
  if (count > max_count) /* found a longer sequence than previous one */
    {
      max_index=i;    /* storing the index of the start element of the longest
                      sequence found so far */
      max_count=count; /* storing the number of elements of the longest
                         sequence found so far */
      i=j+1;          /* start looking for the next longer one from the
                      (j+1)th element in the next iteration */
    }
}
printf("The most recent longest sequence of successively increasing GDP
                rates:\n");
for (i=max_index; i <= max_index + max_count; ++i)
  printf("%d : %lf%%\n",year[i],arr[i]);
putchar('\n');
/* Part (b) of the question */                                               [3]
for (i=0, max_diff = -1; i<num-1; ++i)
{
diff = arr[i+1] - arr[i];   /* taking difference between successive years */
if (diff < 0) diff = -diff; /* and its absolute value, if negative */
 if (diff > max_diff)
  {
    max_diff = diff;
    max_diff_index = i;
  }
}
printf("Largest difference in GDP growth rates was between %d and %d.\n",
year[max_diff_index], year[max_diff_index+1]);
printf("%d : %lf%%\n%d : %lf%%\n",year[max_diff_index],arr[max_diff_index],yea
r[max_diff_index+1],arr[max_diff_index+1]);
printf("Difference in GDP rates : %lf%%.\n",arr[max_diff_index+1]-
arr[max_diff_index]); putchar('\n');
}
```

4. *Completing* `bitcount` *function:*

a. (i) `x != 0` (or) `x > 0` (ii) `b++` **[2]**

b. This is done in order to pad the right-shifted number with 0s, regardless of the sign bit. **[1]**

5. *Predicting the output:* **[9]**

a. **Compile-time error** since the statement `e = c+d = b*a` is illegal. L-value required.

b. p1 points to j and p2 points to i. j has14 and i has10;
The statement `i = i + j * i;` is equivalent to `i = i + (j * i`); hence i becomes 150. Therefore, the output is **164**

c. `(b, a++)` will be take on the value of the left operand of the comma operator, i.e., it will be 5. The left side of the `||` operator will be true. `(a=0)` will not be evaluated due to short circuiting, and hence b will be assigned 1. Therefore output is **6 1**

d. In `foo()` local is modified and global i is increased to 11. Therefore output will be

11

11

e. `a > b >c` is equivalent to (`a > b) > c`. The result is false.

f. a is (111100) b is (001101) a&b = (001100) =12; a|b = (111101) = 61. Output is **12 61**

6. *Pattern-printing question:*

(a) `is (j==0 || j==N-1 || i==j || i+j == N-1).` **[2]**

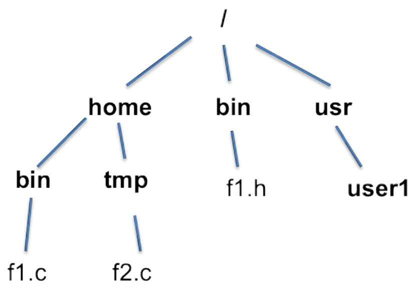
(b) `is printf("\n").` **[1]**

7. *Brief answers to questions:*

a. `rm *.[!c]` **[1]**

b. `grep "^[A-Za-z].*[^0-9]$"` **[1]**

c. The tree structure is as follows: **[2]**



d. Integer division is being performed, which results in truncation of the answer, which is also an integer (and then stored in a float). The situation can be rectified by declaring `sum` as a float or typecasting it to a float. **[1]**

e. Size of `pch` = size of `pshort` = size of `pdouble`. Pointers store addresses, notwithstanding what they point to, and hence are of the same size. **[1]**

f. Order of evaluation of the operands of an operator (except four) is not specified by the language, and is compiler-dependent. Hence, the results vary from compiler to compiler. It is best to avoid such statements. **[2]**

Answer to bonus-credit question: **[2]**
When dereferencing a pointer to `short int`, the compiler accesses `sizeof(short)` bytes, even if the address of a char variable had been stored in the short int pointer. For instance, the GCC accesses data for 2 bytes (and not 1), and hence prints a value other than 10.