Indian Institute of Technology Kharagpur
Programming and Data Structures (CS10001)
Autumn 2017-18: Class Test 1

Time: 1 Hr. **Full Marks**: 20

Section:		**Roll:**			**Name:**			
Q. 01	Q. 02	Q. 03	Q. 04	Q. 05	Q. 06	Q. 07	Q. 08	**Total**

Please write the answers within the boxes provided in questions 1 through 6. Fill up the marked blanks in questions 7 and 8. Any answer written elsewhere will not be evaluated.

---

1. What is the output of the following C Program? **[1 + 0.5 * 2 = 2]**

```
#include <stdio.h>

int main() {
    int a = 3, b = 7, c, d;
    double e;

    c = a + b / 4 * 4.0;
    d = a + b / 4.0 * 4.0;
    e = a + (double)b / 4 * 4;
    printf("%d %d %lf\n", c, d, e);

    return 0;
}
```

2. Write the values for the following expressions: **[0.5 * 2 = 1]**

   (a) `3 + 4 * - 5 / 2` (b) `z = z = 2 && 3 > 4`

3. In the context of **[1 + 1 = 2]**

```
int a[] = {19, 23, 35, 47, 11};
```

   if `&a[1]` is 100, then write the element at memory address 108 and address of array element having value 19. It is given that a variable of `int` type occupies 4 bytes.

   (a) Array element at memory address 108:

   (b) Address of array element having value 19:

4. In the context of expressions `E1`, `E2`, `E3` and block of statements `B`, represent the given `for` loop in terms of a `while` loop. **[1]**

```
for(E1; E2; E3)
    B;
```

5. What is the output of the following C Program? **[2]**

```
# include <stdio.h>
int main() {
    int i;
    for (i = 0; i < 20; i++) {
        switch (i) {
            case 0: i += 5;
            case 1: i += 2;
            case 5: i += 5;
            default: i += 4; break;
        }
        printf("%d ", i);
    }
    return 0;
}
```

6. Consider the following C program. Write the output for every input shown in the table. Hint: Matching `else` with `if` and understanding the nesting may help. **[0.5 * 4 = 2]**

```
#include <stdio.h>

int main() {
int x = 0, n;
scanf("%d", &n);

if (n > 8) {
if (n < 9) x = 1;
}
else {
if (n > 4)
if (n < 6) x = 2;
else x = 3;
else
if (n > 2) x = 4;
else x = 5;
}
printf("%d\n", x);
return 0;
}
```

Input	Output
1	
3	
7	
9	

7. Let `A` be an array containing positive integers. The following C program computes the length of the longest increasing sequence of numbers in `A`. For example, for

   `A = {1, 9, 2, 5, 8, 6, 4}`

   the increasing sequences of length `1` are {`1`}, {`9`}, {`2`}, {`5`}, {`8`}, {`6`}, {`4`}; of length `2` are {`1, 9`}, {`2, 5`}, {`5, 8`}; and of length `3` are {`2, 5, 8`}. There is no increasing sequence of length `4` or higher. So the length of the longest increasing sequence is `3`. Hence the result is `3`.

   Complete the program by filling up the missing lines in the code. **[5 * 1 = 5]**

```
#include <stdio.h>

int main() {
    int A[] = { 2, 3, 5, 2, 3, 4, 7, 9, 1, 3, 2, 5, 7, 9 }; // Input array
    const int n = 14;    // Number of elements in the array
    int i = 0;           // Index variable to iterate over the array
    int len = 1;         // Length of the current increasing sequence
    int maxlen = 1;      // Length of the longest increasing sequence so far

    while (               ) {   // Iterate on array A. Note that the last
           ---------------      // element does not have a next element

        if (               )    // Check if the sequence is increasing
            ---------------

                                // Update current increasing
            ---------------     // sequence length

        else {
            if (len > maxlen)   // Update length of the
                maxlen = len;   // longest sequence so far

                                // Re-init (Reset) current
            ---------------     // increasing sequence length
        }

                                // Move to next element
        ---------------
    }
    if (len > maxlen)    // Update length of the
        maxlen = len;    // longest sequence so far

    printf("Length of longest increasing sequence = %d\n", maxlen);

    return 0;
}
```

8. A Dudeney number is a positive integer that equals the cube of its sum of digits. For example, 512 = 8 * 8 * 8 is a Dudeney number because 8 = 5 + 1 + 2. Similarly, 4913 = 17 * 17 * 17 is Dudeney as 17 = 4 + 9 + 1 + 3.

   Complete the following C program to check if a given number is a Dudeney number.

   [5 * 1 = 5]

```
#include <stdio.h>

int main() {
    int n;        // Input number
    int m;        // Copy of input number n
    int i;        // Index variable to extract digits
    int s = 0;    // Variable to accumulate the sum of digits of n
    int d;        // Next digit in m from right to left scan

    scanf("%d", &n);    // Read n
    m = n;              // Copy n to m

    // Extract and accumulate sum of digits
    for (i = 0;               ; ++i) {
                ____________    // Termination of loop

        d =                     // Find the next digit in m
            ____________        // from right to left scan

        s =                     // Accumulate the sum of digits in s
            ____________

        m =                     // New m after removing
            ____________        // the right-most digit d
    }

    // Check for Dudeney number condition and print

    if (                           )
        ____________________________

        printf("%d is a Dudeney number\n", n);
    else
        printf("%d is not a Dudeney number\n", n);

    return 0;
}
```

1 Solutions:

1. `7 10 10.000000`
2. `-7, 0`
3. `47, 96`
4.

```
E1;
while (E2) {
    B;
    E3;
}
```

5. `16 21`
6. `5, 4, 3, 0`

7.

```
#include <stdio.h>

int main() {
    int A[] = { 2, 3, 5, 2, 3, 4, 7, 9, 1, 3, 2, 5, 7, 9 }; // Input array
    const int n = 14;    // Number of elements in the array
    int i = 0;           // Index variable to iterate over the array
    int len = 1;         // Length of the current increasing sequence
    int maxlen = 1;      // Length of the longest increasing sequence so far

    while (i < n - 1      ) {   // Iterate on array A. Note that the last
           ---------------      // element does not have a next element

        if (A[i] < A[i + 1])    // Check if the sequence is increasing
            ---------------

            ++len;              // Update current increasing
            ---------------     // sequence length

        else {
            if (len > maxlen)   // Update length of the
                maxlen = len;   // longest sequence so far

            len = 1;            // Re-init (Reset) current
            ---------------     // increasing sequence length
        }

        ++i;                    // Move to next element
        ---------------
    }
    if (len > maxlen)    // Update length of the
        maxlen = len;    // longest sequence so far

    printf("Length of longest increasing sequence = %d\n", maxlen);

    return 0;
}
```

8.

```
#include <stdio.h>

int main() {
    int n;        // Input number
    int m;        // Copy of input number n
    int i;        // Index variable to extract digits
    int s = 0;    // Variable to accumulate the sum of digits of n
    int d;        // Next digit in m from right to left scan

    scanf("%d", &n);    // Read n
    m = n;              // Copy n to m

    // Extract and accumulate sum of digits
    for (i = 0; m              ; ++i) {
                ---------------    // Termination of loop

        d = m % 10;                // Find the next digit in m
            ---------------        // from right to left scan

        s = s + d;                 // Accumulate the sum of digits in s
            ---------------

        m = m / 10;                // New m after removing
            ---------------        // the right-most digit d
    }

    // Check for Dudeney number condition and print
    if (s*s*s == n     )
        ---------------

        printf("%d is a Dudeney number\n", n);
    else
        printf("%d is not a Dudeney number\n", n);

    return 0;
}
```