Indian Institute of Technology, Kharagpur

Department of Computer Science and Engineering

Mid-Semester Examination, Autumn 2013-14

Programming and Data Structures (CS 11001)

Students: 700 **Date**: 30-Sep-13 (FN)

Full marks: 60 **Time**: 2 hours

Name	**Roll No.**	**Section**

Question No.	1	2	3	4	5	6	**Total**
Marks Obtained							

01

Instructions:

1. Write your name, roll number and section in the space above.
2. On the top of every odd page write your roll number.
3. Answer all questions in the space provided in this paper itself. No extra sheet will be provided.
4. Use the designated spaces and the last sheet for rough work.
5. Marks for every question is shown with the question.
6. Questions have been shuffled across different question papers.
7. No further clarifications to any of the questions will be provided.

1. Answer the following questions:

   (a) Convert 5c.a3 (in hexadecimal form) to the binary number system. [**2 Marks**]

   ```
   Answer: 1011100.10100011
   ```

   (b) Convert 209.375 (in decimal system) to the hexadecimal system. [**2 Marks**]

   ```
   Answer: d1.6
   ```

(c) Represent 53 and -69 in binary 8-bit 2's complement representation. [**1 + 1 = 2 Marks**]

```
 53 =

-69 =
```

```
Answer:
 53   = 00110101
-69   = 10111011
```

(d) The following expression has a single pair of parentheses missing that makes it syntactically wrong and it does not compile. Place a pair of parentheses in proper places to correct the expression and evaluate it for `x = 2`, `y = 3`, `z = 5`, `w = 7`: [**2 + 1 = 3 Marks**]

```
x * w + y * z = 7
```

Corrected Expression:

Value of the corrected expression:

```
Answer:
Corrected Expression: x * w + y * (z = 7)
Value of the corrected expression: 35
```

(e) What will be the value of `j` for below-mentioned values of `i`? [**1 + 1 + 1 + 1 = 4 Marks**]

```
switch (i) {
    case 2: i = i * i;
    case 4: i = i * i;
    default: i = i * i;
        break;
    case 16: i = i * i;
}
j = i;
```

```
For i =  2, j = ______

For i =  4, j = ______

For i = 16, j = ______

For i =  1, j = ______
```

```
Answer:
For i =  2, j = 256
For i =  4, j = 256
For i = 16, j = 256
For i =  1, j = 1
```

(f) Write the output from the following program: [**1 + 1 + 2 = 4 Marks**]

```
#include <stdio.h>

int main() {
    int w = 0, x = 2.5, y = 5, z = 3, r, s = 4, t = 5, u = -3;
    double a = 2.36, b = 3.19, c = 3.0, d = 2.91726;

    printf("Expr_1 = %d\n", (int)(c * y / z + y / z * c));
    printf("Expr_2 = %lf\n", x - s * t * - c - u);
    printf("Expr_3 = %f\n",
        (float)(x + y < z + w && a > b - 17 * x || ! x < 5));
    return 0;
}
```

```
Expr_1 = __________

Expr_2 = __________

Expr_3 = __________
```

```
Answer:
Expr_1 = 8
Expr_2 = 65.000000
Expr_3 = 1.000000
```

(g) Write the output of the following program. [**0.5 + 0.5 = 1 Mark**]

```
#include <stdio.h>

#define m 5+5
const int n = 5+5;

void main() {
    int a = 0, b = 0;

    a = m * m;
    b = n * n;
    printf("%d %d\n", a, b);
}
```

```
Answer:
35 100
```

(h) You need to convert the following for-loop into an equivalent while-loop: [**0.5 + 0.5 + 0.5 + 0.5 = 2 Marks**]

```
for(i = 2; i <= sqrt(n); i += 3)
    printf("%d ", i);
```

Fill up the blank lines.

```
____________________;

while (____________________) {

    ____________________;

    ____________________;
}
```

```
Answer:

    i = 2;
    while (i <= sqrt(n)) {
        printf("%d ", i);
        i += 3;
    }
```

(i) What will be the output from the following program? [**2 Marks**]

```
#include <stdio.h>
void f(int a[], int n)
{
    int i;
    for (i = 0; i < n - 1; ++i)
        a[i] += a[i+1];
}

int main ()
{
    int a[5] = {1, 2, 4, 6, 8};
    f(a, 4);
    printf("%d", a[4] - a[3]);
    return 0;
}
```

```
Answer: 2
```

(j) What will be the output from the following program? [$\mathbf{1 + 1 + 1 + 1 = 4}$ **Marks**]

```
#include <stdio.h>

int main()
{
    int i = 1, j = 1, k = 1, count = 0;

    while (i < 2) {
        for(; j < 4; j += k)
            do {
                ++count;
                k += i;
            } while (k < 8);
        i += j;
    }

    printf("Loop Indices: %d %d %d\n", i, j, k);
    printf("Number of iterations = %d\n", count);

    return 0;
}
```

```
Loop Indices: _____ _____ _____

Number of iterations = _____
```

```
Answer:
Loop Indices: 10 9 8
Number of iterations = 7
```

(k) Write the output of the following program. [**2 Marks**]

```
#define n 5

void main() {
    int a[n];
    int i;

    a[0] = 1;
    for(i = 0; i < n - 1; ++i) {
        a[a[i]] = a[i]+1;
    }

    for(i = 0; i <= n - 1; ++i) {
        printf("%d ", a[i]);
    }
}
```

```
Answer:
1 2 3 4 5
```

2. Write the output from the following program: [**5 Marks**]

```
#include <stdio.h>

void main() {
    int a[] = {22, 19, 17, 36, 12, 15, 28, 35, 66, 43};
    int i, j, n = sizeof(a)/sizeof(int);

    for(i = 0; i < n; ++i)
        for(j = 0; j < i; ++j)
            if (a[i] > a[j]) {
                a[i] = a[i] + a[j];
                a[j] = a[i] - a[j];
                a[i] = a[i] - a[j];
            }

    for(i = 0; i < n; ++i)
        printf("%d ", a[i]);
    printf("\n");
}
```

```
Answer:
66 43 36 35 28 22 19 17 15 12
```

3. Consider the following program:

```
#include <stdio.h>

unsigned int h(unsigned int n) {
    if (0 == n)
        return 0;
    else
        return h(n/2) + n % 2;
}

int main() {
    unsigned int nMax = 16, sum = 0, n = 0;

    for(; n < nMax; ++n) {
        sum += h(n);
    }
    printf("Sum = %u\n", sum);
    return 0;
}
```

(a) Compute `h(n)` for `n = 2, 5,` and `7` to get an idea for what the function `h(n)` does. Describe `h(n)` in words for a given `n`. [**1 + 1 + 1 + 1 = 4 Marks**]

```
h(2) = _____

h(5) = _____

h(7) = _____

h(n) = ____________________________________________________________
```

```
Answer:
h(2) = 1
h(5) = 2
h(7) = 3
h(n) = number of 1's in the binary representation of n.
```

(b) What will be the output of the above program? [**3 Marks**]

```
Answer:
Sum = 32
```

(c) What will be the output of the program if `nMax` is initialized to 2^K for some $K > 0$? [**3 Marks**]

Answer:

$K2^{K-1}$

4. The following program intends to compute π by Newton's formula upto the 10000-th term:

$$\frac{\pi}{2} = \sum_{k=0}^{\infty} \frac{2^k (k!)^2}{(2k+1)!}$$

Fill up the dashed lines in the program. [**0.5 + 0.5 + 0.5 + 3 + 0.5 = 5 Marks**]

Note: Since factorial of a number cannot be computed in a single expression, you need to find an iterative formula to compute the k-th term t_k from the $(k-1)$-st term t_{k-1}.

```
#include <stdio.h>
int main ()
{
    double sum = _____;

    double term = _____;

    int k = _____;

    for(; k < 10000; ++k) {

        term = ________________________________________;


        sum = ____________________;
    }

    sum *= 2;

    printf("%lf", sum);
    return 0;
}
```

Answer:

```
#include <stdio.h>

int main_pi()
{
    double sum = 1.0;
    double term = 1.0;
    int k = 1;

    for(; k < 10000; ++k) {
        term = term*((double)k/(double)(2*k+1));
        sum = sum + term;
    }
    sum *= 2;

    printf("%lf", sum);
    return 0;
}
```

5. The following function Solve() is designed to solve a quadratic equation $ax^2 + bx + c = 0$. Hence it takes three floating-point (real) coefficients as input parameters a, b & c, and generates up to two roots (wherever possible) of the equation. Solve() sets the roots in the global variables r1 & r2. Further Solve() returns a status value retVal to describe the type of the solution computed. The status can take the following values:

Return Value	**Description**
Sol_Inconsistent	Equation is inconsistent and there is no solution
Sol_Linear	Equation is linear and there is one real root
Sol_RepeatedRoots	Equation has repeated real roots
Sol_RealRoots	Equation has distinct real roots
Sol_ComplexRoots	Equation has complex conjugate roots
Sol_Infinite	Equation has infinite roots

```
const int Sol_Inconsistent = 0;
const int Sol_Linear = 1;
const int Sol_RepeatedRoots = 2;
const int Sol_RealRoots = 3;
const int Sol_ComplexRoots = 4;
const int Sol_Infinite = 5;

double r1; // Root 1
double r2; // Root 2
```

```
int Solve(double a, double b, double c)
{
    if (0 == a) {
        if (0 == b) {
            if (0 == c) { // Infinite solutions
                retVal = Sol_Infinite;
            } else {
                retVal = _________________________;
            }
        } else {
            retVal = _________________________;

            r1 = _________________________;

            r2 = _________________________;
        }
    } else {
        double disc = b*b - 4*a*c;
        if (0 == disc) {

            retVal = _________________________;

            r1 = _________________________;

            r2 = _________________________;
        } else {
            if (disc > 0) {
                retVal = _________________________;

                r1 = _________________________;

                r2 = _________________________;
            } else { // Complex conjugate roots
                retVal = Sol_ComplexRoots;
                // r1 and r2 need not be set in this case
            }
        }
    }

    return retVal;
}
```

Fill up the blank lines in the code above [**0.5*10 = 5 Marks**]

```
Answer:

int Solve(double a, double b, double c)
{
    unsigned int retVal = 0;
    if (0 == a) {
        if (0 == b) {
            if (0 == c) { // Infinite solutions
                retVal = Sol_Infinite;
            } else { // Inconsistent equation
                retVal = Sol_Inconsistent;
                         -----------------
            }
        } else { // Linear equation
            retVal = Sol_Linear;
                     ----------
            r1 = -c/b;
                 ----
            r2 = -c/b;
                 ----
        }
    } else {
        double disc = b*b - 4*a*c;
        if (0 == disc) { // Repeated roots
            retVal = Sol_RepeatedRoots;
                     -----------------
            r1 = -b/(2*a);
                 --------
            r2 = -b/(2*a);
                 --------
        } else {
            if (disc > 0) { // Real distinct roots
                retVal = Sol_RealRoots;
                         -------------
                r1 = (-b + sqrt(disc))/(2*a);
                     -----------------------
                r2 = (-b - sqrt(disc))/(2*a);
                     -----------------------
            } else { // Complex conjugate roots
                retVal = Sol_ComplexRoots;
                // ...
            }
        }
    }

    return retVal;
}
```

6. Consider the following C structure for representing a vector $\vec{v} = (v_x, v_y)$ in two-dimensional Cartesian coordinate system:

```
typedef struct Vector_tag {
    double x;
    double y;
} Vector;
```

The following code segment computes the norm $\sqrt{(v_x^2 + v_y^2)}$ of a vector $\vec{v}$:

```
#include <stdio.h>

#include ____________________

double norm(Vector v) {

    return ________________________________________;
}
```

The following function computes the difference vector $(\vec{v}_1 - \vec{v}_2)$ of two vectors $\vec{v}_1$ and $\vec{v}_2$:

```
Vector diff(Vector v1, Vector v2) {
    Vector v;

    v.x = ____________________;

    v.y = ____________________;

    return v;
}
```

(a) Fill up the missing lines above. [**1 + 1 + 0.5 + 0.5 = 3 Marks**]

```
Answer:

#include <stdio.h>
#include <math.h>
         --------

double norm(Vector v) {
    return sqrt(v.x*v.x + v.y*v.y);
           -----------------------
}

Vector diff(Vector v1, Vector v2) {
    Vector v;
    v.x = v1.x - v2.x;
          -----------
    v.y = v1.y - v2.y;
          -----------
    return v;
}
```

(b) Fill up the gap below to compute the Euclidean distance between two points p and q by suitably calling the functions `norm(v)` and `diff(v1, v2)`: [**2 Marks**]

```
Vector p, q;
double dist;
// ...

dist = ________________________________________; // distance from p to q
```

```
Answer:

dist = norm(diff(p, q));
       ----------------
```

(c) The following function takes a vector $\vec{v} = (v_x, v_y)$ as input and outputs its reflection $\vec{v}_r = (-v_x, -v_y)$ about the origin $(0, 0)$.

```
______________ reflect(Vector v) {
    Vector vr;

    ______________________________;

    ______________________________;
    return vr;
}
```

Complete the function [$\mathbf{1 + 0.5 + 0.5 = 2}$ **Marks**]

```
Answer:

Vector reflect(Vector v) {
------
    Vector vr;

    vr.x = -v.x;
    -----------
    vr.y = -v.y;
    -----------
    return vr;
}
```