Bismillahir Rahmanir Rahim

International Islamic University Chittagong

Department of Computer Science & Engineering

Mid Term Examination, Spring 2023

CSE-1121 Computer Programming I

[Solution]

1(a)

(i) Not equal
(ii) In C language, floating point numbers can't be stored accurately since the numbers aren't stored in binary format.
(iii) Let a & b be two double variables. So, we can check the equality of a & b as follows:

```
if (fabs(a - b) < 0.00001) printf("Equal");
else printf("Not equal");
```

1(b)

```
double p = -3.78e-8; // e can be in either upper or lower case
```

or,

```
double p = -0.0000000378;
```

1(c)

(i)
```
area = sqrt(s * (s - a) * (s - b) * (s - c));
```

(ii)
```
x = (-b - sqrt(b * b - 4 * a * c)) / (2 * a);
```

1(c) OR

(i) One
Two
(ii) None

1(d)

```
#include <stdio.h>
int main () {
    double x, fx;
    int a, b, c, d;
    scanf("%d%d%d%d%lf", &a, &b, &c, &d, &x);
    fx = a * x * x * x + b * x * x + c * x + d;
    printf("f(x) = %lf", fx);
    return 0;
}
```

1(e)

Given,
a = 10,
b = 5,

	c = 0, x = 0.01, y = 0.05, d = ‘d’, t = ‘2’ **(1)** c * a + b % a + c => **0 * 10** + 5 % 10 + 0 => 0 + **5 % 10** + 0 => **0 + 5** + 0 => **5 + 0** => 5 **(2)** b + x + a * 3.5 - y || d && t % b++ < a => 5 + 0.01 + **10 * 3.5** - 0.05 || ‘d’ && ‘2’ % 5++ < 10 => **5 + 0.01** + 35.0 - 0.05 || ‘d’ && ‘2’ % 5++ < 10 => **5.01 + 35.0** - 0.05 || ‘d’ && ‘2’ % 5++ < 10 => **40.01 - 0.05** || ‘d’ && ‘2’ % 5++ < 10 => 39.96 || ‘d’ && ‘2’ % 5++ < 10 => 1 [Since, the left hand side of the || operator is truthy, the result of this expression is truthy i.e. 1. In this way, the right hand side of the || operator isn’t evaluated at all. This behavior is called Short-circuit evaluation.] **(3)** x + a * b - -y + d * t % b => 0.01 + 10 * 5 - **-0.05** + ‘d’ * ‘2’ % 5 => 0.01 + **10 * 5** - -0.05 + ‘d’ * ‘2’ % 5 => 0.01 + 50 - -0.05 + **‘d’ * ‘2’** % 5 => 0.01 + 50 - -0.05 + **5000 % 5** [ASCII(‘d’) = 100, ASCII(‘2’) = 50] => **0.01 + 50** - -0.05 + 0 => **50.01 - -0.05** + 0 => **50.06 + 0** => 50.06
2(a)	**(i)** For n = 5, the first if-condition is true. The second if-condition is also true. So, x /= 9 => x = x / 9 => x = 5 / 9 => **x = 0** and, y += 2 => y = y + 2 => y = AA + 2 The expression **y-= 5** isn’t attached to any if/else statement. So, this expression will be evaluated with any value of n. So, y = y - 5

	=> y = AA + 2 - 5 **=> y = AA - 3** Finally, **x = 0, y = AA - 3**. **(ii)** For n = 3, the first if-condition is true but the second isn't. So, the else block is executed. So, x *= 3 => x = x * 3 => x = 5 * 3 => x = 15 and, y -= 5 => y = y - 5 => y = AA - 5 Finally, **x = 15, y = AA - 5**. **(iii)** For n = -1, the first if-condition is false. So, neither the second if nor the else block is executed. Thus x remains unchanged. and, y -= 5 => y = y - 5 => y = AA - 5 Finally, **x = 5, y = AA - 5**.

2(b)	**#include <stdio.h>
	int main() {
	int N, A, B, C;
	double result;
	scanf ("%d", &N);
	switch (N) {
	case 1:
	scanf ("%d%d", &A, &B);
	result = 2.0 * (A + B);
	printf ("%.2lf\n", result);
	break;
	case 2:
	scanf ("%d%d%d", &A, &B, &C);
	result = ((A + B) * C) / 2.0;
	printf ("%.2lf\n", result);
	break;
	case 3:
	scanf ("%d", &A);
	result = 3.14159 * A * A / 4.0;**

```
            printf ("%.2lf\n", result);
            break;
    }
    return 0;
}
```

2(c)

```
#include <stdio.h>
int main() {
    int mathA, mathB, phyA, phyB, sumA, sumB;
    scanf ("%d%d%d%d", &mathA, &phyA, &mathB, &phyB);
    sumA = mathA + phyA;
    sumB = mathB + phyB;
    if (sumA > sumB) printf ("A\n");
    else if (sumA < sumB) printf("B\n");
    else {
        if (mathA > mathB) printf ("A\n");
        else if (mathA < mathB) printf ("B\n");
        else printf ("BOTH\n");
    }
    return 0;
}
```

2(c) OR

```
#include <stdio.h>
int main() {
    int h1, h2, b1, b2, a1, a2;
    scanf ("%d%d%d%d", &h1, &b1, &h2, &b2);
    a1 = h1 * b1;
    a2 = h2 * b2;
    if (a1 > a2) printf ("T1\n");
    else if (a1 < a2) printf("T2\n");
    else {
        if (h1 > h2) printf ("T1\n");
        else if (h1 < h2) printf ("T2\n");
        else printf ("SAME\n");
    }
    return 0;
}
```

3(a)

(i)
2
1
0

(ii)
3
2
1
0

3(b)

```
#include <stdio.h>
int main() {
    int n, i, a, p = 1;
    scanf ("%d", &n);
    for (i = 1; i <= n; i++) {
        scanf ("%d", &a);
        p *= a;
        printf("%d\n", p);
    }
    return 0;
}
```

3(c)

```
#include <stdio.h>
int main() {
    int n, i, a, three = 0, five = 0, total, sum;
    scanf ("%d", &n);
    for (i = 1; i <= n; i++) {
        scanf ("%d", &a);
        if (a == 3) three++;
        else if (a == 5) five++;
    }
    total = three + five;
    sum = three * 3 + five * 5;
    printf ("%d\n%d\n", total, sum);
    return 0;
}
```

3(c) OR

```
#include <stdio.h>
int main() {
    int n, i, a, sum = 0, count = 0;
    double avg;
    scanf ("%d", &n);
    for (i = 1; i <= n; i++) {
        scanf ("%d", &a);
        if (a % 7 == 0) {
            sum += a;
            count++;
        }
    }
    avg = (sum * 1.0) / count;
    printf ("%.2lf\n", avg);
    return 0;
}
```