

Name:

Roll No:

For Evaluators only

Total: / 12

Evaluator:

1. What is the output of the following program?

```
1  #include <stdio.h>
2  int temp = 0;
3  int fun(int x, int y) {
4      int z;
5      temp++;
6      if (y == 3) return (x * x);
7      else {
8          z = fun(x, y / 3);
9          return (z * z);
10     }
11 }
12 int main() {
13     int c = fun(2, 81);
14     printf("%d %d", temp, c);
15     return 0;
16 }
```

Answer: 4, 65526 (2^{16})

2. What should you fill in the single line marked as TODO?

```
1  #include "stdio.h"
2  typedef struct Node { int data; struct Node* next; } Node;
3  typedef Node* LinkedList;
4  int element_at(int pos, LinkedList l) {
5      // TODO: write single line which returns element
6      // at position `pos` in `l`
7      // Solution:
8      return pos == 0 ? l->data : element_at(pos-1, l->next);
9  }
10 int main() {
11     Node third = {22, NULL};
12     Node second = {26, &third};
13     Node first = {20, &second};
14     LinkedList l = &first;
15     printf("%d\n", element_at(1, l)); // should print 26
16     printf("%d\n", element_at(2, l)); // should print 22
17     return 0;
18 }
```

3. What is the output of the following program?

```
1  #include <stdio.h>
2  int main() {
3      int a[] = {1, 2, 3, 4, 5};
4      int *p = a;
5      printf("%d ", *p++);
6      printf("%d", *(p + 1));
7      return 0;
8  }
```

Answer: 1 3

4. What is the output of Cal(10, 8)?

```
1  void Cal(int a, int b) {
2      if (b != 1) {
3          if (a != 1) {
4              printf("*");
5              Cal(a / 2, b);
6          } else {
7              b = b - 1;
8              Cal(10, b);
9          } } }
```

Answer: prints 21 times *

5. What is the output?

```
1  #include <stdio.h>
2  void fun(int **ptr2, int **ptr1) {
3      int *ii;
4      ii = *ptr2;
5      *ptr2 = *ptr1;
6      *ptr1 = ii;
7      **ptr1 = **ptr2;
8      **ptr2 += **ptr1;
9  }
10 void main( ) {
11     int a=5, b=10;
12     int *p=&a, *q=&b;
13     fun(&p, &q);
14     printf("%d %d", a, b);
15 }
```

Answer: 50 60

6. Write an expression in C language which uses only operators $+$, $-$, $\&$, \wedge , $<$, $>$ and variable identifiers x , y , which computes the minimum of x and y .

Answer:

$$y + ((x - y) \& ((x - y) >> 31));$$

7. What is the output?

```
1 #include <stdio.h>
2 int main() {
3     char str[] = "Hello, World!";
4     printf("%s", str + 8);
5     return 0;
6 }
```

Answer: orld!

8. What is the output?

```
1 #include <stdio.h>
2 int main() {
3     int x = 5;
4     int y = x << 2;
5     printf("%d", y);
6     return 0;
7 }
```

Answer: 20

9. What is the output?

```
1 enum {false,true};
2 int main() {
3     int i=1;
4     do {
5         printf("%d ",i);
6         i++;
7         if(i < 15) continue;
8     } while(false); // false == 0
9     return 0;
10 }
```

Answer: 1

10. What will happen on running the following program?

```
1 #include <stdio.h>
2 int main() {
3     int a[] = {5, 4, 2, 1, 3};
4     int *p = a;
5     int i = 0;
6     do {
7         printf("%d ", *p);
8         p = a + *p - 1;
9     } while (p != a);
10    return 0;
11 }
```

Answer: 5 3 2 4 1

11. What is the output?

```
1 int foobar(int* n){
2     *n = *n +1;
3     return *n;
4 }
5 // code below inside main function
6 int k = 16;
7 printf("foobar(k) = %d", foobar(&k) );
8 printf(" k = %d\n", k);
```

Answer: foobar(k) = 17, k = 17

12. Write a recursive single-line function logic at the line mentioned TODO, to compute the number of all arrangements of k items from n objects.

```
1 int count_arrangements(int n, int k) {
2     // TODO
3     // Solution:
4     return k == 0 ? 1 : n * count_arrangements(n-1, k-1);
5 }
```

C program code for EndSem theory exam.

```
1 // problem 1
2 // #include<stdio.h>
3 // int temp = 0;
4 // int fun(int x, int y) {
5 //     int z;
6 //     temp++;
7 //     if (y == 3) return (x * x);
8 //     else {
9 //         z = fun(x, y / 3);
```

```

10 //      return (z * z);
11 //    }
12 // }
13 // int main() {
14 //     int c = fun(2, 81);
15 //     printf("%d %d", temp, c);
16 //     return 0;
17 // }

18
19 // Problem 2
20 // #include "stdio.h"
21 // typedef struct Node { int data; struct Node* next; } Node;
22 // typedef Node* LinkedList;
23 // int element_at(int pos, LinkedList l) {
24 //     // TODO: write single line which returns element
25 //     // at position `pos` in `l`
26 //     // Solution:
27 //     return pos == 0 ? l->data : element_at(pos-1, l->next);
28 // }
29 // int main() {
30 //     Node third = {22, NULL};
31 //     Node second = {26, &third};
32 //     Node first = {20, &second};
33 //     LinkedList l = &first;
34 //     printf("%d\n", element_at(1, l)); // should print 26
35 //     printf("%d\n", element_at(2, l)); // should print 22
36 //     return 0;
37 // }

38
39
40 // Problem 3
41 // #include <stdio.h>
42 // int main() {
43 //     int a[] = {1, 2, 3, 4, 5};
44 //     int *p = a;
45 //     printf("%d ", *p++);
46 //     printf("%d", *(p + 1));
47 //     return 0;
48 // }

49
50 // Problem 4
51 // void Cal(int a, int b) {
52 //     if (b != 1) {
53 //         if (a != 1) {
54 //             printf("* ");
55 //             Cal(a / 2, b);
56 //         } else {
57 //             b = b - 1;
58 //             Cal(10, b);
59 //         }
60 //     }
61 //     int main() {
62 //         Cal(10, 8);
63 //         return 0;
64 //     }
65 // }

66 // Problem 5
67 // #include <stdio.h>
68 // void fun(int **ptr2, int **ptr1) {
69 //     int *i;
70 //     i = *ptr2;
71 //     *ptr2 = *ptr1;
72 //     *ptr1 = i;
73 //     **ptr1 = **ptr2;
74 //     **ptr2 += **ptr1;
75 // }
76 // void main() {
77 //     int a = 5, b = 10;
78 //     int *p = &a, *q = &b;
79 //     fun(&p, &q);
80 //     printf("%d %d", a, b);
81 // }

82
83 // Problem 6
84 // Write a expression in C language which uses only
85 // operators +, -, &, ^, <, > and variable identifiers x,
86 // y, which computes the minimum of x and y.
87 // #include <stdio.h>
88 // int min1(int x, int y) {
89 //     return y + ((x - y) & ((x - y) >> 31));
90 // }

91 // int min(int x, int y) {
92 //     return ((x & y) + ((x ^ y) & -(x < y)));
93 // }
94 // int main() {
95 //     printf("%d\n", min1(5, 3)); // should print 3
96 //     printf("%d\n", min(3, 5)); // should print 3
97 //     return 0;
98 // }

99
100 // Problem 7
101 // #include <stdio.h>
102 // int main() {
103 //     char str[] = "Hello, World!";
104 //     printf("%s", str + 8);
105 //     return 0;
106 // }

107
108 // Problem 8
109 // #include <stdio.h>
110 // int main() {
111 //     int x = 5;
112 //     int y = x << 2;
113 //     printf("%d", y);
114 //     return 0;
115 // }

116
117 // Problem 9
118 // int main() {
119 //     int i = 1;
120 //     do {
121 //         printf("%d ", i);
122 //         i++;
123 //         if (i < 15) continue;
124 //     } while (0);
125 //     return 0;
126 // }

127
128
129 // Problem 10
130 // #include <stdio.h>
131 // int main() {
132 //     int a[] = {5, 4, 2, 1, 3};
133 //     int *p = a;
134 //     int i = 0;
135 //     do {
136 //         printf("%d ", *p);
137 //         p = a + *p - 1;
138 //     } while (p != a);
139 //     return 0;
140 // }

141
142
143 // Problem 11
144 // #include <stdio.h>
145 // int foobar(int* n) {
146 //     *n = *n + 1;
147 //     return *n;
148 // }
149 // int main() {
150 //     int k = 16;
151 //     printf("foobar(k) = %d", foobar(&k));
152 //     printf("k = %d\n", k);
153 // }

154
155
156 // Problem 12,
157 // #include <stdio.h>
158 // int count_arrangements(int n, int k) {
159 //     // TODO
160 //     // Solution:
161 //     return k == 0 ? 1 : n * count_arrangements(n-1, k-1);
162 // }
163 // int main() {
164 //     printf("%d\n", count_arrangements(5, 3)); // should print 60
165 //     printf("%d\n", count_arrangements(5, 5)); // should print 120
166 //     return 0;
167 // }

168
169
170

```