

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 +, -, &, ^, <, > 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?

Answer: 5 3 2 4 1

```
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 }
```

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

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

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

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

78 // Problem 6
79 // Write a expression in C language which uses only
80 // operators +, -, *, ^, <, > and variable identifiers x,
81 // y, which computes the minimum of x and y.
82 // #include <stdio.h>
83 // int min1(int x, int y) {
84 //     return y + ((x - y) & ((x - y) >> 31));
85 // }
86 // int min(int x, int y) {
87 //     return ((x & y) + ((x ^ y) & -(x < y)));
88 // }
89 // int main() {
90 //     printf("%d\n", min1(5, 3)); // should print 3
91 //     printf("%d\n", min(3, 5)); // should print 3
92 //     return 0;
93 // }

94 // Problem 7
95 // #include <stdio.h>
96 // int main() {
97 //     char str[] = "Hello, World!";
98 //     printf("%s", str + 8);
99 //     return 0;
100 // }

101 // Problem 8
102 // #include <stdio.h>
103 // int main() {
104 //     int x = 5;
105 //     int y = x << 2;
106 //     printf("%d", y);
107 //     return 0;
108 // }

109 // Problem 9
110 // int main() {
111 //     int i=1;
112 //     do {
113 //         printf("%d ", i);
114 //         i++;
115 //         if(i < 15) continue;
116 //     } while(0);
117 //     return 0;
118 // }

119 // Problem 10
120 // #include <stdio.h>
121 // int main() {
122 //     int a[] = {5, 4, 2, 1, 3};
123 //     int *p = a;
124 //     int i = 0;
125 //     do {
126 //         printf("%d ", *p);
127 //         p = a + *p - 1;
128 //     } while (p != a);
129 //     return 0;
130 // }

131 // Problem 11
132 // #include <stdio.h>
133 // int foobar(int* n){
134 //     *n = *n + 1;
135 //     return *n;
136 // }
137 // int main(){
138 //     int k = 16;
139 //     printf("foobar(k) = %d,", foobar(&k) );
140 //     printf(" k = %d\n", k);
141 // }

142 // Problem 12,
143 // #include <stdio.h>
144 // int count_arrangements(int n, int k) {

```

```
160 // // TODO
161 // // Solution:
162 // return k == 0 ? 1 : n * count_arrangements(n-1, k-1);
163 // }
164 // int main() {
165 //     printf("%d\n", count_arrangements(5, 3)); // should print 60
166 //     printf("%d\n", count_arrangements(5, 5)); // should print 120
167 //     return 0;
168 // }
169
170
```
