Computer Programming — End Sem Theory: Solutions

Instructor: Girish Varma • Course Code: CS0.101 • IIIT Hyderabad — 2 hrs exam · 12 questions 1 mark each

Name: Roll No:

For Evaluators only

Total: / 12 Evaluator:

1. What is the output of the following program?

```c
#include < stdio.h >
int temp = 0;
int fun(int x, int y) {
    int z;
    temp++;
    if (y == 3) return (x * x);
    else {
        z = fun(x, y / 3);
        return (z * z);
    }
}
int main() {
    int c = fun(2, 81);
    printf("%d %d", temp, c);
    return 0;
}
```

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

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

```c
#include "stdio.h"
typedef struct Node { int data; struct Node* next; } Node;
typedef Node* LinkedList;
int element_at(int pos, LinkedList l) {
    // TODO: write single line which returns element
    // at position `pos` in `l`
    // Solution:
    return pos == 0 ? l->data : element_at(pos-1, l->next);
}
int main() {
    Node third = {22, NULL};
    Node second = {26,&third};
    Node first = {20,&second};
    LinkedList l = &first;
    printf("%d\n", element_at(1,l)); // should print 26
    printf("%d\n", element_at(2,l)); // should print 22
    return 0;
}
```

3. What is the output of the following program?

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

Answer: 1 3

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

```c
void Cal(int a, int b) {
    if (b != 1) {
        if (a != 1) {
            printf("*");
            Cal(a / 2, b);
        } else {
            b = b - 1;
            Cal(10, b);
} } }
```

Answer: prints 21 times $*$

5. What is the output?

```c
#include<stdio.h>
void fun(int **ptr2,int **ptr1) {
        int*ii;
        ii=*ptr2;
        *ptr2=*ptr1;
        *ptr1=ii;
        **ptr1 *= **ptr2;
        **ptr2 += **ptr1;
}
void main( ) {
        int a=5, b=10;
        int *p=&a, *q=&b;
        fun(&p,&q);
        printf("%d %d",a, b);
}
```

Answer: 50 60

6. Write a expression in C language which uses only operators $+, -, \&, \hat{}, <, >$ 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?

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

Answer: orld!

8. What is the output?

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

Answer: 20

9. What is the output?

```c
enum {false,true};
int main() {
    int i=1;
    do {
        printf("%d ",i);
        i++;
        if(i < 15) continue;
    } while(false); // false == 0
    return 0;
}
```

Answer: 1

10. What will happen on running the following program?

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

Answer: 5 3 2 4 1

11. What is the output?

```c
int foobar(int* n){
     *n = *n +1;
     return *n;
}
// code below inside main function
int k = 16;
printf("foobar(k) = %d,",foobar(&k) );
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.

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

C program code for EndSem theory exam.

```c
// problem 1
// #include<stdio.h>
// int temp = 0;
// int fun(int x, int y) {
//     int z;
//     temp++;
//     if (y == 3) return (x * x);
//     else {
//         z = fun(x, y / 3);
```

```
//         return (z * z);
//     }
// }
// int main() {
//     int c = fun(2, 81);
//     printf("%d %d", temp, c);
//     return 0;
// }

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


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

// Problem 4
// void Cal(int a, int b) {
//     if (b != 1) {
//         if (a != 1) {
//             printf("* ");
//             Cal(a / 2, b);
//         } else {
//             b = b - 1;
//             Cal(10, b);
// } } }
// int main() {
//     Cal(10, 8);
//     return 0;
// }


// Problem 5
// #include<stdio.h>
// void fun(int **ptr2,int **ptr1) {
//         int*ii;
//         ii=*ptr2;
//         *ptr2=*ptr1;
//         *ptr1=ii;
//         **ptr1 *= **ptr2;
//         **ptr2 += **ptr1;
// }
// void main( ) {
//         int a=5, b=10;
//         int *p=&a, *q=&b;
//         fun(&p,&q);
//         printf("%d %d",a, b);
// }

// Problem 6
// Write a expression in C language which uses only
// operators +, , &, ^, <, > and variable identifiers x,
// y, which computes the minimum of x and y.
// #include <stdio.h>
// int min1(int x, int y) {
//     return y + ((x - y) & ((x - y) >> 31));
// }
// int min(int x, int y) {
//     return ((x & y) + ((x ^ y) & -(x < y)));
// }
// int main() {
//     printf("%d\n", min1(5, 3)); // should print 3
//     printf("%d\n", min(3, 5)); // should print 3
//     return 0;
// }

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

// Problem 8
// #include <stdio.h>
// int main() {
//     int x = 5;
//     int y = x << 2;
//     printf("%d", y);
//     return 0;
// }


// Problem 9
// int main() {
//     int i=1;
//     do {
//         printf("%d ",i);
//         i++;
//         if(i < 15) continue;
//     } while(0);
//     return 0;
// }


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


// Problem 11
// #include <stdio.h>
// int foobar(int* n){
//      *n = *n +1;
//      return *n;
// }
// int main(){
//     int k = 16;
//     printf("foobar(k) = %d,",foobar(&k) );
//     printf(" k = %d\n", k);
// }


// Problem 12,
// #include <stdio.h>
// int count_arrangements(int n, int k) {
//   // TODO
//   // Solution:
//   return k == 0 ? 1 : n * count_arrangements(n-1, k-1);
// }
// int main() {
//   printf("%d\n", count_arrangements(5, 3)); // should print 60
//   printf("%d\n", count_arrangements(5, 5)); // should print 120
//   return 0;
// }
```