1. Find the output (operator and expression)

c

#**include** <stdio.h>

**int** main() {

**char** s = '\123'; *// '\123' is octal for 83*

printf("%d", (**int**)**sizeof**(s));

**return** 0;

}

Output:  
1

1. Find the output (functions)

c

#**include** <stdio.h>

**int** main() {

**static** **int** i = 3;

printf("%d", i--);

**return** (i > 0) ? main() : 0;

}

Output:  
321

1. Find the output (pointers)

c

#**include** <stdio.h>

**int** main() {

**char** \*s[] = {"dharmr'a", "hewlett-packard", "siemens", "ibm"};

**char** \*\*p;

p = s;

printf("%s", ++\*p); *// harmr'a*

printf("%s", \*p++); *// harmr'a*

printf("%s", ++\*p); *// ewlett-packard*

**return** 0;

}

Output:  
harmr'aharmr'aewlett-packard

1. Find the output (dynamic memory)

c

#**include** <stdio.h>

#**include** <stdlib.h>

#**include** <string.h>

**int** main() {

**int** i;

**char** a[] = "String";

**char** \*p = "New String";

**char** \*temp;

temp = malloc(strlen(p) + 1);

strcpy(temp, p);

p = malloc(strlen(temp) + 1);

strcpy(p, temp);

printf("%s", p);

free(temp);

free(p);

**return** 0;

}

Output:  
New String

1. Find the output (algorithm)

c

#**include** <stdio.h>

**int** main() {

**int** n = 12, res = 1;

**while**(n > 3) {

n -= 3;

res \*= 3;

}

printf("%d", n \* res);

**return** 0;

}

Output:  
81

1. Find the output (function)

c

#**include** <stdio.h>

**void** fun(**int** b[][3]) {

*// ++b; // Not valid, pointer arithmetic on 2D arrays is tricky*

b[1][1] = 5;

}

**int** main() {

**int** a[3][3] = { {9,8,7}, {6,5,4}, {3,2,1} };

fun(a);

printf("%d\n", a[2][1]);

**return** 0;

}

Output:  
2

1. Find the output (strings)

c

#**include** <stdio.h>

#**include** <string.h>

**int** main() {

**int** i, n;

**char** x[5];

strcpy(x, "Zoho");

n = strlen(x);

strcpy(x, x + (n - 1));

printf("%s", x);

**return** 0;

}

Output:  
o

1. Find the output (arrays)

c

#**include** <stdio.h>

**int** main() {

**int** c[] = {5,4,3,4,5};

**int** j, \*q = c;

**for**(j = 0; j < 5; j++) {

printf("%d", \*q);

++q;

}

**return** 0;

}

Output:  
54345

1. Find the output (branching and looping)

c

#**include** <stdio.h>

**int** main() {

**int** i = -1;

**for**(i = 0; i > -1; i--) {

printf("%d", i);

**if**(i != -1) **break**;

}

**return** 0;

}

Output:  
0

1. Find the output (arrays)

c

#**include** <stdio.h>

**int** main() {

**int** s[] = {1,0,5,0,10,0};

**int** f[] = {2,4,6,8,10,12};

**int** n = 6, i = 0, j = 0;

**for**(j = 1; j < n; j++) {

**if**(s[j] > -f[i]) {

printf("%d", i);

i = j;

}

}

**return** 0;

}

Output:  
0

1. Find the output (branching and looping)

c

#**include** <stdio.h>

**int** main() {

**int** i = 0, j = 0, sum = 0;

**for**(i = 1; i < 500; i += 3)

**for**(j = 0; j < i; j++)

sum++;

printf("%d", sum);

**return** 0;

}

Output:  
20825

1. Find the output (branching and looping)

c

#**include** <stdio.h>

**int** main() {

**int** n;

**for**(n = 6; n != 1; n--) {

printf("%d", n--);

}

**return** 0;

}

Output:  
642

1. Find the output (arrays)

c

#**include** <stdio.h>

**int** main() {

**int** a[3][4] = {

{2,4,6,5},

{10,12,12,10},

{5,6,4,2}

};

**int** i = 0, j, k = 99;

**while**(i < 3) {

**for**(j = 0; j < 4; j++) {

**if**(a[i][j] < k)

k = a[i][j];

}

i++;

}

printf("%d", k);

**return** 0;

}

Output:  
2

1. Find the output (pointer)

c

#**include** <stdio.h>

#**include** <string.h>

**int** main() {

**char** \*x = "Alice";

**int** i, n = strlen(x);

\*x = x[n];

**for**(i = 0; i < n; i++) {

printf("%s", x); x++;

}

printf("\n");

**return** 0;

}

Output:  
Segmentation fault (runtime error)

1. Find the output (structures and union)

c

#**include** <stdio.h>

**struct** value {

**int** bit1:1;

**int** bit3:4;

**int** bit4:4;

} bit;

**int** main() {

printf("%zu\n", **sizeof**(bit));

**return** 0;

}

Output:  
4

1. Find the output (dynamic memory)

c

#**include** <stdio.h>

#**include** <stdlib.h>

**struct** node {

**int** data;

**float** d;

**struct** node \*link;

};

**int** main() {

**struct** node \*p, \*q;

p = (**struct** node \*) malloc(**sizeof**(**struct** node));

q = (**struct** node \*) malloc(**sizeof**(**struct** node));

printf("%zu,%zu\n", **sizeof**(p), **sizeof**(q));

free(p);

free(q);

**return** 0;

}

Output:  
8,8

1. Find the output (structures and unions)

c

#**include** <stdio.h>

#**include** <string.h>

**typedef** **union** {

**int** a;

**char** b[10];

**float** c;

} Union;

**int** main() {

Union x, y = {100};

x.a = 50;

strcpy(x.b, "Hello");

x.c = 21.50;

printf("%d %s %f\n", x.a, x.b, x.c);

printf("%d %s %f\n", y.a, y.b, y.c);

**return** 0;

}

Output:  
Unpredictable (union overwrites memory), but typically:  
Unpredictable Hello 21.500000  
100 (garbage) 0.000000

1. Find the output (structures and union)

c

#**include** <stdio.h>

**struct** point {

**int** x;

**int** y;

} origin, \*pp;

**int** main() {

pp = &origin;

printf("origin is (%d %d)\n", (\*pp).x, (\*pp).y);

printf("origin is (%d %d)\n", pp->x, pp->y);

**return** 0;

}

Output:  
origin is (0 0)  
origin is (0 0)

1. Find the output (branching and looping)

c

#**include** <stdio.h>

**int** main() {

**int** i = -1;

printf("%d %d\n", i, +i);

**return** 0;

}

Output:  
-1 -1

1. Find the output (datatypes)

c

#**include** <stdio.h>

**int** main() {

**char** not;

not = 12;

printf("%d", not);

**return** 0;

}

Output:  
12

1. Find the output (branching and looping)

c

#**include** <stdio.h>

#**define** FALSE -1

#**define** TRUE 1

#**define** NULL 0

**int** main() {

**if**(NULL)

puts("NULL");

**else** **if**(FALSE)

puts("TRUE");

**else**

puts("FALSE");

**return** 0;

}

Output:  
TRUE

1. Find the output (operator and expressions)

c

#**include** <stdio.h>

**int** main() {

**int** k = 1;

printf("%d-1 is %s", k, k-1 ? "TRUE" : "FALSE");

**return** 0;

}

Output:  
1-1 is TRUE

1. Find the output (file manipulation)

c

#**include** <stdio.h>

**int** main() {

FILE \*ptr;

**char** i;

ptr = fopen("demo.c", "r");

**while**((i = fgetc(ptr)) != EOF)

printf("%c", i);

fclose(ptr);

**return** 0;

}

Output:  
Prints the contents of demo.c file (if exists)

1. Find the output (branching and looping)

c

#**include** <stdio.h>

**int** main() {

**int** t, i;

**for**(t = 4; scanf("%d", &i) - t; printf("%d\n", i))

printf("%d--", t--);

**return** 0;

}

Output:  
Loop runs 4 times

1. Find the output (structures and unions)

c

#**include** <stdio.h>

#**include** <stdlib.h>

#**include** <string.h>

**struct** emp {

**int** len;

**char** name[1];

};

**int** main() {

**char** newname[] = "Rahul";

**struct** emp \*p = (**struct** emp \*) malloc(**sizeof**(**struct** emp) - 1 + strlen(newname) + 1);

p->len = strlen(newname);

strcpy(p->name, newname);

printf("%d %s\n", p->len, p->name);

free(p);

**return** 0;

}

Output:  
5 Rahul

1. Find the output (algorithm)

c

#**include** <stdio.h>

**int** main() {

printf("%d %d %d %d\n", 72, 072, 0x72, 0X72);

**return** 0;

}

Output:  
72 58 114 114

1. Find the output (operator and expression)

c

#**include** <stdio.h>

**int** main() {

**char** ch;

**int** a;

**float** b;

printf("bytes occupied by ch-%zu\n", **sizeof**(ch));

printf("bytes occupied by a-%zu\n", **sizeof**(a));

printf("bytes occupied by b-%zu\n", **sizeof**(b));

**return** 0;

}

Output:  
bytes occupied by ch-1  
bytes occupied by a-4  
bytes occupied by b-4

1. Find the output (operator and expressions)

c

#**include** <stdio.h>

**int** main() {

printf("%zu\n", **sizeof**('7'));

printf("%zu\n", **sizeof**(7));

printf("%zu\n", **sizeof**(7.0));

**return** 0;

}

Output:  
4  
4  
8

1. Find the output (datatypes)

c

#**include** <stdio.h>

**int** main() {

**char** ch = 35; *// 291 overflows char, 35 = '#'*

printf("%d %d %c\n", -2147483648, ch, ch);

**return** 0;

}

Output:  
-2147483648 35 #

1. Find the output (datatypes)

c

#**include** <stdio.h>

**int** main() {

**int** g;

g = 300000 \* 300000 / 300000;

printf("g-%d\n", g);

**return** 0;

}

Output:  
g-300000

1. Find the output (datatypes)

c

#**include** <stdio.h>

**int** main() {

**float** a;

a = 4 / 2;

printf("%f %f\n", a, 4 / 2);

**return** 0;

}

Output:  
2.000000 2.000000

1. Find the output (operator and expression)

c

#**include** <stdio.h>

**int** main() {

printf("%zu\n", **sizeof**(4) / **sizeof**(2.0));

printf("%zu\n", **sizeof**(2.0) / **sizeof**(4));

**return** 0;

}

Output:  
0  
2

1. Find the output (operator and expression)

c

#**include** <stdio.h>

**int** main() {

**int** x = 10, y = 5, p, q;

p = x > 9;

q = x > 3 && y != -3;

printf("p-%d q-%d\n", p, q);

**return** 0;

}

Output:  
p-1 q-1