GATE

MADE EASY Workbook 2026

Detailed Explanations of Try Yourself *Questions*

Computer Science & IT

Programming
and Data Structures

MADE EASY PUBLICATIONS

1 Programming

Detailed Explanation of Try Yourself Questions

T1 : Solution

1. Const int *P;
   declare P as pointer to const integer.
2. int * const P;
   declare P as constant pointer to integer

T2 : Solution

(*i*) Char (*(*x ( ))[ ])();
declare x as a function returning pointer to array of pointer to function returning char.

(*ii*) Char (*(*x[3])( ) [5];
declare x as array 3 of pointer to function returning pointer to array 5 of char.

(*iii*) Void (*b*int, void (*f)(int))) (int);
Syntac error

(*iv*) Void (*ptr)(int (*)[2], int(*)(void));
Syntax error

T3 : Solution

(b)

Char \ 0
if (0) ∵∵∵ Printf(% S", a) = Null = 0
So condition false
So answer is else part string is not empty.

T4 : Solution

(c)

Take random value of X and Y i.e., X = 5 and Y = 3.

Initially X = 5, Y = 3, res = 1, a =X and b =Y.

Option (a): $X^Y = a^b$

$$X^Y = a^b \equiv 5^3 = 5^3 \equiv 125 = 125$$

After iteration 1

$$\text{res} = 5;\ a = 5;\ b = 2;\ X = 5;\ Y = 3$$
$$X^Y = a^b \equiv 5^3 \neq 5^2 \equiv 125 \neq 25$$

So, case fail. Option (a) cannot be answer.

Option (b): $(\text{res} * a)^Y = (\text{res} * X)^b$

$$(1 \times 5)^3 = (1 \times 5)^3 \equiv 125 = 125$$

After iteration 1

$$\text{res} = 5;\ a = 5;\ b = 2;\ X = 5;\ Y = 3$$
$$(\text{res} * a)^Y = (\text{res} * X)^b \equiv (5 \times 5)^3 = (5 \times 5)^2$$
$$15625 \neq 625$$

So, case fail. Option (b) cannot be answer.

Option (d): $X^Y = (\text{res} * a)^b$

$$5^3 = (1 \times 5)^3 \equiv 125 = 125$$

After iteration 1

$$\text{res} = 5;\ a = 5;\ b = 2;\ X = 5;\ Y = 3$$
$$X^Y = (\text{res} * a)^b \equiv 5^3 = (5 \times 5)^2$$

$125 \neq 625$ So, case fail.

Option (d) cannot be answer.

Option (c): $X^Y = \text{res} * a^b$

$$5^3 = 1 \times 5^3 \equiv 125 = 125$$

After iteration 1

$$\text{res} = 5;\ a = 5;\ b = 2;\ X = 5;\ Y = 3$$
$$X^Y = \text{res} * a^b \equiv 5^3 = 5 \times 5^2 \equiv 125 = 125$$

After iteration 2

$$\text{res} = 25;\ a = 5;\ b = 1;\ X = 5;\ Y = 3$$
$$X^Y = \text{res} * a^b \equiv 5^3 = 25 \times 5^1 \equiv 125 = 125$$

So, all cases are passes.

So option (c) will the answer.

T5 : Solution

(d)

Definition of anagram: a string s is a string obtained by permuting the letter in s.

1st for loop just create an array count and initialize to 0 for each element i.e. for 128 element.

While loop run till string reaches to end of strings. In while loop, for every element $a[i]$ count is increased which is decreased by array $b[i]$ when same element arrive in array $b[i]$ and increment loop with j++. If both strings are anagram of each other the count value will be zero for each elements.

So, count[$a[j]$] ++; and count[$b[j$++]] – – ;

Example: a =

		g	a	t	e	/n

b =

a	t	g	e	/n

Count	0	0	0	0						
	✗ Ø	Ø ✗	✗ Ø	Ø ✗	0	0	0	0	0	0
	a	t	g	e	ASCII (128)					

Inside while loop

(1) count[g]++;
count[a]– –;
j++;

(2) count[a]++;
count[t]– –;
j++;

(3) count[t]++;
count[g]– –;
j++;

(4) count[e]++;
count[e]– –;
j++;

While loop fail for (5).

Since atlast count will contain all zero. So both are anagram of each other.

T6 : Solution

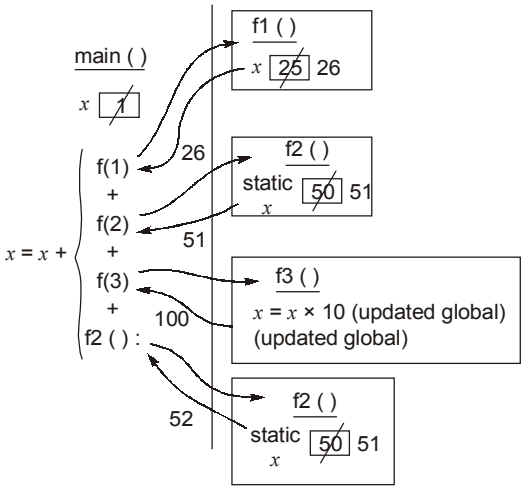
(b)

x = 15
fun (5, &x) let &x = 100
↓
fun (5, 100)
↓ t = 5, f = 5 + 3, *f_p = 5
return 8
fun (4, 100)
↓ t = 3, f = 3 + 2, *f_p = 3
return 5
fun (3, 100)
↓ t = 2, f = 2 + 1, *f_p = 2
return 3
fun (2, 100)
↓ t = 1, f = 1 + –1, *f_p = 1
return 2
fun (1, 100) *f_p = 1
return 1

So fun(5, & x) will **return 8** and it will be printed.

T7 : Solution

(230)



$x = 1 + 26 + 51 + 100 + 52 = 230$

T8 : Solution

(b)

While calculating ${}^{n}C_3 = \dfrac{n(n-1)(n-2)}{6}$ i.e. multiplying n × $(n-1) \times (n-2)$ then divide by 6.

Since n value can be large such that $n \times (n-1) \times (n-2)$ will generate value which cannot be representable by bits assigned to unsigned int.

So, idea is to calculate ${}^{n}C_3$ as $\left(\dfrac{n(n-1)}{2}\right)\dfrac{(n-2)}{3}$.

Since $n(n-1)$ always gives an even number greater than 0, so always divisible by 2 but if $n(n-1)$ divide by 3 first then it may be possible that answer comes in floating point, since we use integer variable so part after decimal point need to be discarded i.e. will not get exact answer as ${}^{n}C_3$.

So most appropriate choice is

$$= \left(\frac{n(n-1)}{2}\right) \times \frac{(n-2)}{3}$$

T9 : Solution

(c)

Here $x == (y \times q + r)$

q = Quotient and r = Remainder

In given code indirectly we are dividing 'r' by 'y' in form of substraction and 'q' count the number of time number gets divide successfully before reaching $r \geq y$.

So, initial condition will be $q = 0$ i.e., $r = x$ and final condition $(q == 0)$ && $(r == x)$ && $(y > 0)$.

T10 : Solution

(d)

In given program we take the test cases and apply

From T_1 if all values are equal means a = b = c = d, a = b condition satisfied. S_1 and S_4 executed.

From T_2 when all a, b, c, d distinct:

S_1, S_2 not execute, S_3 executes.

From T_3 when a = b, S_1 execute but c = d.

S_2 will not execute but S_3 and S_4 executes.

Here no need of T_3 because we get all result from above two.

From T_4 if a! = b and c = d, S_1 not executes and S_2 and S_4 executes.

All of S_1, S_2, S_3, S_4 will execute and covered by T_1, T_2 and T_4.

T11 : Solution

(d)

10101101 i.e. option (d)

Since recursive function calls will be

$$1^{st} \text{ function call } \frac{173}{2} = 86(\text{int part only})$$

$$2^{nd} \text{ function call } \frac{86}{2} = 43$$

$$3^{rd} \text{ function call } \frac{43}{2} = 21(\text{int part only})$$

$$4^{th} \text{ function call } \frac{21}{2} = 10(\text{int part only}),$$

$$5^{th} \text{ function call } \frac{10}{2} = 5$$

$$6^{th} \text{ function call } \frac{5}{2} = 2 \text{ (integer part only)},$$

$$7^{th} \text{ function call } \frac{2}{2} = 1$$

In 7^{th} function call condition if(n <=1) will become true so n will be printed (i.e. 1 will be printed) now while returning every function will execute its remaining part of code i.e., printf ("%d", n%2);

So 6^{th} function will print 2 mod 2 = 0,

5^{th} function will print 5 mod 2 = 1,

4^{th} function will print 10 mod 2 = 0,

3^{rd} function will print 21 mod 2 = 1,

2^{nd} function will print 43 mod 2 = 1,

1^{st} function will print 86 mod 2 = 0,

Function call f(173) will print

173 mod 2 = 1

T12 : Solution

(b)

$$\text{Address of a [40][50]} = \text{Base address} + 40 \times 100 \times \text{Element size} + 50 \times \text{Element size}$$
$$= 0 + 4000 \times 1 + 50 \times 1 = 4050$$

Therefore, option (b) is correct.

T13 : Solution

(2)

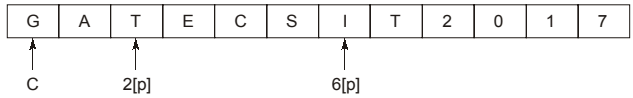
(C + 2[p] – 6[p] – 1)

C + 'T' – 'I' – 1

= C + 11 – 1

= C + 10

Hence printf will print '2'.



■■■■

2 Linked List, Stack, Queue and Hashing

T1 : Solution

Implementation of stack using single link list:

Inserting sequence: 1, 2, 3, 4, 5, 6

Insertion take 0(1) time

Link list representation:

1. → [1 | /]
2. → [2 |] → [1 | /]
3. → [3 |] → [2 |] → [1 | /]
4.
5.
6. Insertion takes 0(1) time.

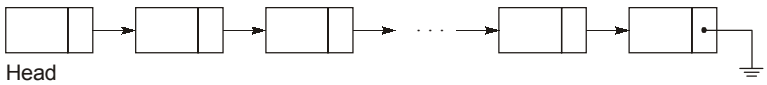
Deletion in stack (Pop)

Remove top element every time so 0(1)

Deletion in link list

Remove 1st node every time with making second node to head.

T2 : Solution



enqueue operation takes $O(1)$ time

dequeue operation takes $O(n)$ time [visits last node]

T3 : Solution

(d)

PUSH (S, P, Q, T_i, x)
{
  if $\left(T_i == \left(\frac{P}{Q} \times (i+1) - 1\right)\right)$
  {
    printf ("stack overflow");
    exit (1);
  }
  else
    T_i++;
  $S[T_i] = x$;
}

$T_i == \left(\frac{P}{Q} \times (i+1) - 1\right)$ indicate the last location of the array is already filled. So overflow occur.

T4 : Solution

(22079)

Formula to find location of $a[20][20][30] = 10 + \{[(20-1)(30-1)(40-1)] + (20-1)(30-1) + (30-1)\}$

$= [10 + (19 \times 29 \times 39) + (19 \times 29) + (29)]$

$= 10 + 21489 + 551 + 29$

$= 10 + 22069$

$= 22079$

T5 : Solution

(0.7324)

Expected number of probes in a unsuccessful $= \frac{1}{(1-\alpha)}$

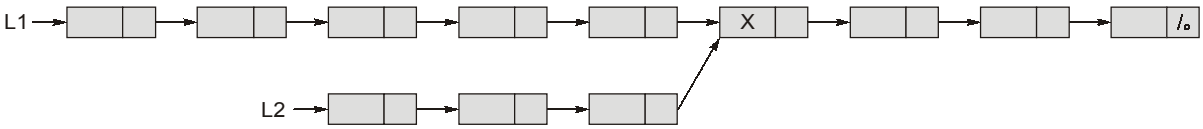
$$\frac{1}{1-\alpha} = 3$$
$$1 = 3(1-\alpha)$$
$$1 = 3 - 3\alpha$$
$$-2 = -3\alpha$$
$$\alpha = \frac{2}{3}$$

Expected number of probes in a unsuccessful $= 1/\alpha \log_e 1/(1-\alpha)$

$$\frac{3}{2}\log_e 3 = 0.7324$$

T6 : Solution

(b)



We need to traverse both the linked list of size m and n.

So it will take $O(m + n)$.

T7 : Solution

(b)

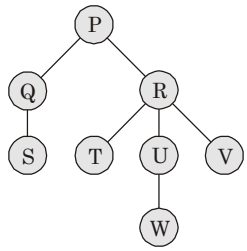
By using BSF (Breadth First Search) traversal we can set the twin pointer in each entry in each adjacency list.
So it will take $\Theta(m + n)$ times (since adjacency list are using).

■■■■

3 Trees

T1 : Solution

(a)



Inorder : (Left, root, mid, right)

∴ **Inorder is :** S Q P T R W U V

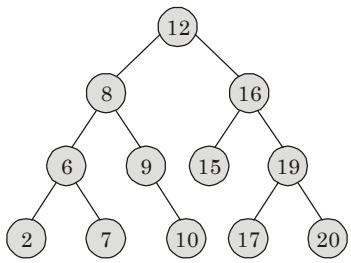
T2 : Solution

(b)

Preorder: 12, 8, 6, 2, 7, 9, 10, 16, 15, 19, 17, 20

Inorder: 2, 6, 7, 8, 9, 10, 12, 15, 16, 17, 19, 20

Tree will be,



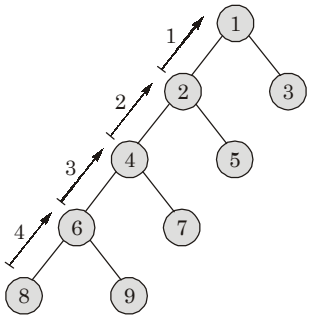
Postorder will be, 2, 7, 6, 10, 9, 8, 15, 17, 20, 19, 16, 12

T3 : Solution

(4)

Postorder = 8, 9, 6, 7, 4, 5, 2, 3, **(1)** Root
Inorder = 8, 6, 9, 4, 7, 2, 5, 1, 3

Binary tree will be



So, height of binary tree will be 4.

■■■■

