Programming and Data Structures

Syllabus

Programming in C. Recursion. Arrays, stacks, queues, linked lists, trees, binary search trees, binary heaps, graphs.

Chapter wise Weightage Analysis

Chapter / Paper / Year	Ch.1	Ch.2	Ch.3	Ch.4	Ch.5	Ch.6	Ch.7	Ch.8
2008	0	2	0	4	2	0	13	2
2009	0	0	1	0	0	0	6	2
2010	0	2	0	3	2	0	0	4
2011	0	0	4	1	0	0	3	0
2012	0	0	4	0	0	2	2	0
2013	0	0	0	0	0	2	2	0
2014 (P1)	0	2	0	0	1	0	2	2
2014 (P2)	1	2	2	0	0	0	3	0
2014 (P3)	0	0	1	0	2	2	1	2
2015 (P1)	0	2	2	0	2	0	1	2
2015 (P2)	0	0	2	2	0	2	3	0
2015 (P3)	0	2	2	1	1	1	2	1
2016 (P1)	0	2	2	3	0	2	4	0
2016 (P2)	0	2	0	4	0	1	4	0
2017 (P1)	2	2	2	4	1	0	2	0
2017 (P2)	0	2	2	5	0	1	1	0
2018	0	2	3	3	0	1	1	0
2019	0	2	5	3	0	0	0	0
2020	0	0	2	3	0	0	3	1
2021 (P1)	0	0	3	2	0	1	0	0
2021 (P2)	0	2	1	1	0	0	1	2
2022	2	0	0	3	1	2	1	1
2023	0	0	3	0	1	2	5	1

CHAPTER

1

DATA TYPES AND OPERATORS

Data Types

1. [MCQ] [GATE-2014 : 1M]

Suppose n and p are unsigned int variables in a C program. We wish to set p to n_{C_3}. If n is large, which one of the following statements is most likely to set p correctly?

(a) p = n * (n – 1)*(n – 2)/6;
(b) p = n * (n – 1)/2 * (n – 2)/3;
(c) p = n * (n – 1)/3 * (n – 2)/2;
(d) p = n * (n – 1)*(n – 2)/6.0;

Operators

2. [MCQ] [GATE-2022 : 2M]

What is printed by the following ANSI C program?

```
#include<stdio.h>
int main(int argc, char *argv[]){
char a = 'P';
char b = 'x';
char c = (a & b) + '*';
char d = (a | b) - '-';
char e = (a ^ b) + '+';
printf("%c %c %c\n", c, d, e);
return 0;
}
```

ASCII encoding for relevant characters is given below

A	B	C	…..	Z
65	66	67	…..	90

a	b	c	…..	z
97	98	99	…..	122

*	+	–
42	43	45

(a) z K S (b) 122 75 83
(c) * – + (d) P x +

3. [NAT] [GATE-2017 : 2M]

Consider the following C program

```
#include<stdio.h>
int main ( ) {
int m=10
int n, n1;
n=++m ;
n1=m++;
n– –;
– –n1 ;
n–=n1;
printf( ("%d", n)
return 0 ;
}
```

The output of the program is_____

❑❑❑

ANSWER KEY

1. (b) **2.** (a) **3.** (0 to 0)

SOLUTIONS

1. (b)

We know that

$$n_{C_3} = \frac{n(n-1)(n-2)}{3!}$$

Let n = 50 int size = 2 byte (ranges from 0 to 65535)

Option (A):

$$50 \times 49 \times \frac{48}{6}$$

$\Rightarrow$ Out of range because of cyclic property. No error, but the results is incorrect.

Hence, Option (a) is wrong

Option (D):

Same as above reason option (d) is wrong.

Option (C):

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

n * (n – 1) is not always divisible by 3.

n = 20

n – 1= 19

$\frac{20 \times 19}{3}$ will produce incorrect result.

Hence, (c) is wrong.

Option (B):

$$\frac{n(n-1)}{2}$$

Product of any 2 successive number is always even (because out of these 2 numbers 1 must be even.)

Product of 3 consecutive number is always divisible by 3 (3!), Because out of these 3 numbers1 must be multiple of 3.

Scan for Video solution

2. (a)

*	+	–
42	43	45

∴ ASCII code of P is 80

ASCII code of x is 120

char c = (a & b) + '*'

( ) is having more priority

a & b

a – 0 1 0 1 0 0 0 0 – 80

&

b – 0 1 1 1 1 0 0 0 – 120

a & b 0 1 0 1 0 0 0 0 = 80

char c = (a & b) + '*';

80 + '*'

80 + 42

char c = 1 2 2

z

On option elimination we get option (a) as correct option.

char = (a/b) – '–';

a 0 1 0 1 0 0 0 0

OR

b 0 1 1 1 1 0 0 0

0 1 1 1 1 0 0 0 = 120

120 – 45 = 75

char d = 75

K

char e = (a ∧ b) + '+';

0 1 0 1 0 0 0 0

0 1 1 1 1 0 0 0

0 0 1 01 0 0 0 ⇒ 40

40 + 43 = 83

char e = 83

S

∴ z K S is printed Hence, (a) is correct option.

Scan for Video solution

3. (0 to 0)

Taking the snippet of code from program.

$n = ++m; \rightarrow \begin{cases} (i)\ m = m+1\ (\text{will make m as 11}) \\ (ii)\ n = m\ (\text{This will assign 11 to n}) \end{cases}$

$n1 = m++ \rightarrow \begin{cases} n1 = m;\ \text{Assign 11 to n1} \\ m = m+1;\ \text{increment the m by 11} \end{cases}$

m ~~10~~ ~~11~~ 12

m ~~G~~ ~~11~~ ~~10~~ 0

m ~~G~~ ~~11~~ 10

n-- → n will decrease by 1 i.e. n becomes 10

n1→ n1 will decrease by 1 (10)

n – = n1 ⇒ n = n –n1 (this will make n as 0,0 is printed)

∴ 0 is printed

Scan for Video solution

❑❑❑

CHAPTER

2

CONTROL FLOW STATEMENTS

Decision Control Statements

1. [NAT] [GATE-2019 : 2M]

Consider the following C program:

```
# include<stdio.h>
int main (){
float sum = 0.0, j = 1.0, i = 2.0;
while (i / j  >  0.0625){
    j = j + j;
    sum = sum + i/j;
    printf("%f \n", sum);
}
return 0;
}
```

The number of times the variable sum will be printed, when the above program is executed, is _____.

2. [MCQ] [GATE-2017 : 2M]

Consider the C program fragment below which is meant to divide x by y using repeated subtractions. The variables x, y, q and r are all unsigned int.

```
while (r >= y)
{
 r = r – y;
q = q + 1;
}
```

Which of the following conditions on the variables x y, q and r before the execution of the fragment will ensure that the loop terminates in a state satisfying the condition x == (y * q + r) ?

(a) (q==r) &&(r==0)
(b) (x > 0) &&(r==x) &&(y>0)
(c) (q==0) &&(r==x) &&(y>0)
(d) (q==0) &&(y>0)

3. [MCQ] [GATE-2016 : 2M]

The following function computes the maximum value contained in an integer array p[ ] of size n(n ≥ 1).

```
int max ( int * p, int n )
{ int a = 0, b = n – 1;
while (_______ )
{
    if(p[a] ≤ p[b]){a = a + 1;}
    else          { b = b – 1;}
}
return p[a];
}
```

The missing loop condition is

(a) a !=n (b) b != 0
(c) b>(a+1) (d) b !=a

4. [MCQ] [GATE-2016 : 2M]

The following program is to be tested for statement coverage:

```
begin
  if (a==b){S1 ; exit; }
  else if (c==d){S2 ;}
  else {S3 ; exit; }
  S4;
end
```

The test cases T1, T2, T3 and T4 given below are expressed in terms of the properties satisfied by the values of variables a, b, c and d. The exact values are not given.

T1: a, b, c and d are all equal

T2: a, b, c and d are all distinct

T3 : a=b and c !=d

T4 : a !=b and c=d

Which of the test suites given below ensures coverage of statements S1, S2, S3 and S4 ?

(a) T1, T2, T3

(b) T2, T4

(c) T3, T4

(d) T1, T2, T4

5. [MCQ] [GATE-2015 : 2M]

Consider the following pseudo code, where x and y are positive integers.

```
begin
q:=0
r:=x ;
while r≥y do
begin r:=r-y ; q:=q+1 ; end
end
```

The post condition that needs to be satisfied after the program terminates is

(a) {r=qx + y∧ r < y}

(b) {x=qy + r ∧ r < y}

(c) {y=qx + r ∧ 0 < r < y}

(d) {q+1< r –y ∧ y > 0}

6. [NAT] [GATE-2014 : 2M]

Consider the following function

```
double f( double x){
if (abs(x*x-3)<0.01) return x;
else return f(x / 2+1.5 / x);
```

Given a value q (to 2 decimals) such that f(q) will return q : _____.

7. [MCQ] [GATE-2008 : 2M]

Choose the correct option to fill ?1 and ?2 so that the program below prints an input string in reverse order. Assume that the input string is terminated by a newline character

```
void reverse(void) {
    int c;
    if (?1) reverse();
    ?2
}
main() {
printf ("Enter Text") ; printf ("\n");
reverse(); printf ("\n");
}
```

(a) ? 1 is (getchar () ! = '\n')
? 2 is getchar (c);

(b) ? 1 is (c = getchar ()) ! = '\n')
? 2 is getchar (c)

(c) ? 1 is (c ! = '\n')
? 2 is putchar (c);

(d) ? 1 is ((c = getchar ()) ! = '\n')
? 2 is putchar (c);

Switch Statements

8. [NAT] [GATE-2015 : 2M]

Consider the following C program:

```
#include<stdio.h>
int main()
{
int i, j, k = 0;
j=2 * 3 / 4 + 2.0 / 5 + 8 / 5;
k-=--j;
for (i=0; i<5; i++)
{
    switch(i+k)
    {
      case 1:
      case 2: printf("\n%d", i+k);
      case 3: printf("\n%d", i+k);
      default: printf("\n%d", i+k);
    }
  }
   return 0;
}
```

The number of times printf statement executed is ___.

Iterative Statements

9. [MCQ] [GATE-2021: 2M]

Consider the following ANSI C program

```
# include<stdio.h>
int main()
{
   int i, j, count;
   count =0;
   i=0;
   for(j=-3 ; j<=3 ; j++)
   {
     if((j>=0) &&(i++))
        count = count +j ;
   }
   count = count +i ;
   printf ("%d", count ) ;
   return 0 ;
```

Which one of the following options is correct?

(a) The program will not compile successfully

(b) The program will compile successfully and output 10 when executed

(c) The program will compile successfully and output 8 when executed.

(d) The program will compile successfully and output 13 when executed.

10. [MCQ] [GATE-2018 : 2M]

Consider the following C code. Assume that unsigned long int type length is 64 bits.

```
unsigned long int fun (unsigned long int n) {
        unsigned long int i, j=0, sum=0;
        for (i=n ; i>1 ; i=i / 2) j++;
        for ( ;  j>1 ; j = j / 2) sum++;
        return sum;
}
```

The value returned when we call fun with the input 2^{40} is:

(a) 4 (b) 5

(c) 6 (d) 40

11. [NAT] [GATE-2017 : 2M]

The output of executing the following C program is_____.

```
#include <studio.h>
int total (int v){
  static int count = 0;
       while (v){
       count += v&1;
       v >>= 1;
    }
      return count;
 }
void main()
{
    static int x=0;
    int i=5;
     for( ; i>0; i– –){
          x = x+ total (i);
         }
 printf (‘’%d\n‘’, x);
}
```

12. [MCQ] [GATE-2016 : 2M]

The following function computes *X^Y* for positive integers X and Y.

```
int exp (int X, int Y )  {
int res = 1, a = X, b = Y ;
while (b != 0){
       if (b % 2==0)
          { a=a* a ; b = b / 2 ;}
      else { res = res*a;  b = b –1 ;
}
return res;    }
```

Which one of the following conditions is TRUE before every iteration of the loop?

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

(b) $(res * a)^Y = (res * X)^b$

(c) $X^Y = res * a^b$

(d) $X^Y = (res * a)^b$

13. [MCQ] [GATE-2014 : 2M]

Consider the following pseudo code. What is the total number of multiplications to be performed?

```
D = 2
for i = 1 to n do
  for j = i to n do
    for k = j + 1  to n do
      D=D * 3
```

(a) Half of the product of the 3 consecutive integers.

(b) One-third of the product of the 3 consecutive integers.

(c) One-sixth of the product of the 3 consecutive integers.

(d) None of the above

❑❑❑

ANSWER KEY

1.	(5 to 5)	**2.**	(c)	**3.**	(d)	**4.**	(d)
5.	(b)	**6.**	(1.72 to 1.74)	**7.**	(d)	**8.**	(10 to 10)
9.	(b)	**10.**	(b)	**11.**	(23 to 23)	**12.**	(c)
13.	(c)						

SOLUTIONS

1. (5 to 5)

i = 20, j = 1.0

(1) j = 1.0

= i/j > 0.0625

= 2.0/1.0 > 0.0625(True,Printed)

(2) j = 2.0

2.0/2.0 > 0.0625 = 1.0 > 0.0625(True,Printed)

(3) j = 4.0

2.0/4.0 > 0.0625

0.5 > 0.0625(True,Printed)

(4) j = 8.0

2.0/8.0 > 0.0625

0.25 > 0.0625(True,Printed)

(5) j = 16.0

2.0/16.0 > 0.0625

0.125 > 0.0625(True,Printed)

(6) j = 32.0

2.0/32.0 > 0.0625

0.0625 > 0.0625(False,Not printed)

Hence 5 times printed.

Scan for Video solution

2. (c)

Let us first understand the concept before solving.

Assume x = 12, y = 5

Let's assume that we want to divide 12(x) by 5(y)

$2 \rightarrow$ Quotient

$5\overline{)12}$

$\underline{10}$

$2 \rightarrow$ Remainder

Performing above operation by using repeated subtraction method then we subtract 5 two times from 12(x).

r = x = 12, y =5

Initially r =12(everything is remaining as we did not subtract 5 from 12 any number of times).

q = 0, because it denotes that how many times 5 has been subtracted from 12.

Now,

(1) r = 12, y = 5 can we subtract 5 from r? yes, because r is greater than y.

r = 12 – 5 = 7

q = 0 + 1 = 1(1 signifies that we have subtracted 5 one time from 12)

(2) r =7, y = 5 can we subtract 5 from r? yes, because r is greater than y.

r = 7 – 5 = 2

q = 1 + 1 = 2(2 signifies that we have subtracted 5 two time from 12)

(3) r = 2, y = 5 can we subtract 5 from r? no, because r is less than y so we need to stop further operation.

r = 2, q = 2

Now, solving question by option elimination method.

Initially q==0 and r = = x

Before execution of fragment the value of q is zero and value of r is x. which matches with option c.

Scan for Video solution

3. (d)

Let us take 1st case

n = 1 ⇒ a = 0, b = 0

P [ ]: | 10 |

0

a b

This is the maximum element (1), hence loop will not run, if there is single element then loop will not run.

Option (a)

a!=n

0!=1(true)

above condition is true for only one element too. Instead, it has to be returned by return statement.

Hence it is false option.

Option (b)

Let us take 2nd case

n = 2 | 10 | 20 | a = 0, b = 1

0 1

a b

1st iteration:

p[a] ≤ p[b] → true

a = a + 1

(Moving a to right side)

| 10 | 20 |

0 1

a b

Let us take case 3

| 20 | 10 |

0 1

a b

1st iteration:

p[a] ≤ p[b] ⇒ False

b = b –1 ⇒ b = 0

| 20 | 10 |

0 1

a b

Case: 4

| 10 | 20 | 50 | 30 | 2 |

0 1 2 3 4

a ..... a b ..... b

(i) p[0] ≤ p [4] ⇒ false

b = b –1 ⇒ 3

Why?

Any Smaller element cannot be maximum if an element at a[th] index is bigger than element at b[th] index then p[b] cannot be maximum ⇒ so we are moving to the left (skipping this element p[b]).

(ii) p[a] ≤ p [b] ⇒ true

10 < 30

10 cannot be maximum

Skip it by moving.

How ⇒ by a = a + 1

(iii) p[a] ≤ p [b] ⇒ true

20 ≤ 30

Skip 20 by a = a + 1

p[a] ≤ p [b]

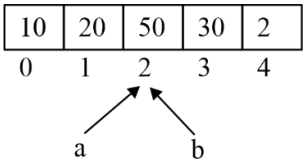
False

50 ≤ 30 ⇒ false

50 can be maximum, But 30 cannot be maximum.

b = b =1

b = 2



*we are left with one element, this must be a maximum element. Now we need not iterate further.

*we need to stop whenever a=b. therefore loop condition will be b!=a or a!=b . i.e. option d is correct answer.

Scan for Video solution

4. (d)

T_1 : S1 T_1, T_2, T_4(covers all)

T_2 : S3 T_2, T_3, T_4(covers all)

T_3 : S1

T_4 : S2, S4

Hence option (d) is correct answer.

Scan for Video solution

5. (b)

Rewriting the code for better understanding

begin q: = 0// q is representing Quotient

r: = x // r is representing Remainder

while r ≥ y do

begin

r: = r – y

q: = q + 1

end

end

The abvoe code is representing, How to divide x by y using repeated subtraction.

Let x = 13, y = 5 and intially q = 0

r = 23 (everyting is remaining)

(i) Can we subtract y(5) from r ?

Yes, r = 13 – 5 = 18

q = q + 1 = 0 +1 =1

r = 8; q = 1 (one time 5 is subtrated from 13)

(ii) r = 8, q = 1 y = 5

can we subtract 5 from r ?

yes, r = 8 – 5 = 3

q = q + 1 = 1 + 1 = 2

r = 3; q = 2 (2 times 5 is subtracted from 13)

(iii) r = 3, q = 2

can we subtract 5 from r ?

No, because $\boxed{x < y}$ which is

Post condition

After code execution

r = 3, q = 2, x = 13 y = 5

$$5\overline{)\,13\,(2}$$
$$\underline{10}$$
$$3$$

we can write, $13 = \underset{2}{q} \times \underset{5}{y} + \underset{3}{r}$

$\boxed{x = qy + r}$ where, r < y

∴ on option verification, option (b) is correct.

Scan for Video solution

6. (1.72 to 1.74)

abs $(x^2 - 3) < 0.01$

absolute value is like an value of mod x

$$|x| = \begin{cases} -x & x < 0 \\ x & x >= 0 \end{cases}$$

$(x^2 - 3) < 0.01$ and $-(x^2 - 3) < 0.01$

$x^2 < 3.01$ and $x^2 > 2.99$

$x < 1.735$ and $x > 1.729$

$x > 1.729$ and $x < 1.735$

$\boxed{x = 1.73}$

Scan for Video solution

7. (d)

(1) option A and option B can be eliminated, because they are not printing anything (No putchar()).

(2) In option (c), there is No getchar() to read input Hence c is wrong.

(3) ∴ correct option is option (d).

Scan for Video solution

8. (10 to 10)

firstly, solving

j=2*3/4+2.0/5+8/5

we know that, *, / having same priority

*, / ↓ high priority
\+ ↓ low priority

and also int, int →int

So, before loop started

$j = \underbrace{2*3/4}_{1} + 2.0/5 + 8/5$

$j = \underset{2}{6/4} + 2.0/5 + 8/5$

$j = \underset{3}{1} + 2.0/5 + 8/5$

2.0 is of double type

5 ⇒ 5.0 (double, promoted implicitly)

Results is also of type double type i.e. 0.4

$j = 1 + 0.4 + \underset{4}{8/5}$

$j = \underbrace{1+0.4}_{5\ \text{double}} + \underset{\text{int}}{1}$

j = 1.4 +1

j = 2.4

j is of int type.

j = 2, 2 is assigned to j

Now,

k = --j;

--j is performed

⇒ j becomes 1

k – = j;

k = 0 –1 = –1

So, before the loop started, $\boxed{k = -1 \text{ and } j = 1}$

Now evaluating loop

```
for (i = 0; i<5; i + +)
{
switch (i + k) {
case 1:
case 2: printf ("\n%d", i +k);
case 3: printf ("\n%d", i +k);
default: printf ("\n%d", i +k);
}
```

i	k	i +k	which case execute	No. of printf
0	–1	–1	only default	1
1	–1	0	only default	1
2	–1	1	case 1, case 2, case 3, default+ No break	3
3	–1	2	case 2,case 3, default	3
4	–1	3	case 3, default	2
loop ends 5				

Number of times printf executed = 1 + 1 + 3 + 3 + 2

= 10 times

Scan for Video solution

9. (b)

Initially variable count and i are initialized to zero.

for(j=-3;j<=3;j++) statement states that, for loop will run from -3 to +3(-3,-2-1,0,1,2,and 3).

Evaluating if condition inside for loop

j = –3

if((j>=0)&&(i++))

–3>=0&&(i++) →false

Here we are using concept of Short Circuiting

Short Circuiting: when the first operand of logical AND operator is 0 then 2nd operand will not be evaluated, if the 1st operand is 1 then 2nd operand will be evaluated.

$$0 \,\&\&\, \underbrace{(i++)}_{\text{will never be evaluated}}$$

That's why

if(0)

count = count + j will not be executed

now, j= -2

if((j>=0)&&(i++))

$$\underbrace{\underbrace{-2>=0}_{\text{False}} \;\&\&\; \underbrace{(i++)}_{\text{will not be evaluated}}}_{(0)\text{false}}$$

0 && (i++)

count = count + j will not be executed because, if(0)(if condition fails) as same as above reason.

j = –1

if((j>=0)&&(i++))

(–1>=0 && (i++))

$$\underbrace{0 \;\&\&\; \underbrace{(i++)}_{\text{will not be evaluated}}}_{\Downarrow\ 0(\text{false})}$$

again count = count +j will not be executed because, if(0)(if condition fails) as same as above reason above mentioned.

j = 0

$$\text{if}\left(\underbrace{0>=0}_{\text{true}} \;\&\&(i++)\right)$$

and becomes

1

$$\left(1 \;\&\; \&\; \underbrace{i++}_{\text{Post Increment}}\right)$$

Post increment: Post increment operator is used when it is required to increment the value of the variable after evaluating the expression.

Initially variable i has value 0.

$\underbrace{1\&\&0}_{(0)\text{false}} \rightarrow$ after evaluation of this statement value of i becomes 1

Hence once again statement count= count + j will not be executed.

j = 1

if(1>=0 && i++)

$\underbrace{1\&\&1}_{(1)\text{ True}} \rightarrow$ after evaluation of this statement value of i becomes 2.

Hence if condition becomes true, so count = count +j will be executed

count = count + j

= 0 + 1

count = 1

j = 2

if(j>=0 &&(i++))

2>=0 && 2

1 && 2

(1) True value of i will become 3

Again, if condition becomes true and statement inside if condition will be executed

count = count + j

= 1 + 2

count = 3

j = 3

if (j>=0 && (i++))

3>=0 && 3

$\underbrace{1}_{(1)\text{ True}} \& \& 1 \rightarrow$ Value of i will become 4

Again, if condition becomes true and statement inside if condition will be executed.

count = count + j

3 + 3

count = 6

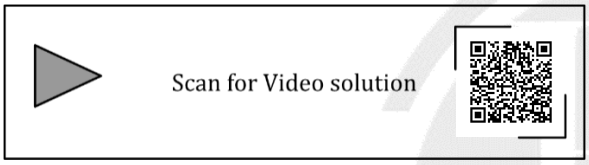
j = 4

condition in for loop gets failed and control of execution comes outside for loop

count=count+i

6 + 4(as value of count is 6 and i is 4)

Hence, count = 10 and lastly 10 is printed by printf statement in the end. Hence option b is correct.



10. (b)

for (i = n; i > 1, i = 1/2)

{

j + 1;

}

Initially j = 0

Let us reduce the input size

$n = 2^6$

i	i>1
2^6	$2^6 > 1 \rightarrow$ True $\rightarrow$ j ++, j = 1
2^5	$2^5 > 1 \rightarrow$ True $\rightarrow$ j ++, j = 2
2^4	$2^4 > 1 \rightarrow$ True $\rightarrow$ j ++, j = 3
2^3	$2^3 > 1 \rightarrow$ True $\rightarrow$ j ++, j = 4
2^2	$2^2 > 1 \rightarrow$ True $\rightarrow$ j ++, j = 5
2^1	$2^1 > 1 \rightarrow$ True $\rightarrow$ j ++, j = 6
i	i>1
2^0	$2^0 > 1 \rightarrow$ Flase

for $n = 2^6$, the first loop

will make j as 6

for $n = 2^{40}$, the loop will make j as 40

Before second loop

j= 40, sum = 0

After second loop

Sum is 5

for (; j > 1; j = j/2)

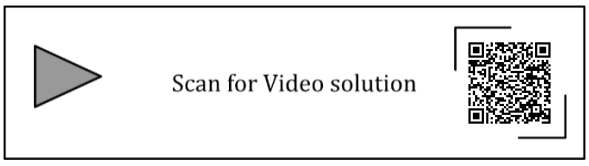
{

sum ++;

}

j	j > 1
40	40 > 1 → True → sum ++
j = j/2 20	20 > 1 → True → sum ++
j = j /2 10	10 > 1 → True → sum ++
j = j/2 5	5 > 1 → True → sum ++
j = j/2 2	2 > 1 → True → sum ++
j = j/2 1	1 > 1 → True → loop ends

After second loop sum is 5.



11. (23 to 23)

(i) **v & 1:** is checking if the rightmost bit of v is 1. (If it is 1, then count value is increment by 1).

(ii) **v>> =1:** Ensuring that every bit is checked.

Because count is a static variable, total(v) is increasing the value of count by the number of

set bits in v and also returns the incremented value.

Let us assume that total (5) is called.

Before total(5)

count | 0 |

5 ⇒ 00000101 (only 2 set bits)

count | ~~0~~ 2 |

count is increased by 2 and returns the same value.

Again, if total (4) is called

Before call count | 2 |

4 → 00000100

count is increased by 1 and same value is returned

count | ~~2~~ 3 |

Now, understanding below code

```
for (; i > 0 ; i--){
x = x + total (1)
}
```

This loop will execute for i = 5, 4, 3, 2, 1 and

x = x + total (i) , x is initially 0

(i) x | 0 |

count | 0 |

i = 5

x = x + total (5)

5 ⇒ 00000101 (2 set bits)

total (5) ⇒ increases the value of count by 2.

count becomes 2 & same value is returned.

x | ~~0~~ 2 |

count | ~~0~~ 2 |

x = x + total

0 + 2

x becomes 2

(ii) x | 2 |

count | ~~2~~ 3 |

i = 4

x = x + total (4)

Binary of 4 contains only one set bit, so it will increase value of count by 1 and same is returned

x | ~~2~~ 5 |

count | ~~2~~ 3 |

x = x + total (4)

2 + 3

x = 5

(iii) i = 3

x | ~~5~~ 10 |

count | ~~3~~ 5 |

x = x + total (3)

Binary of 3 contains 2 one's (00000011) that's why count is increased by 2 i.e. count becomes 5 and is reutned.

x = x + total (3)

x = 5 + 5

x = 10

(iv) i = 2

x | ~~10~~ 16 |

count | ~~3~~ 6 |

→ 00000010

x = x + total (2)

Binary of two contains only one set value so count is increased by 1, count becomes 6 and same is returned.

x = x + total (2)

10 + 6

x = 16

(v) i = 1

x 23

count 7

00000001

x = x + total (1)

Binary of 1 contains only one set value so count is increased by1, count becomes 7 and same is returned,

x = x+ total (1)

16 + 7

x = 23

Finally loop terminates and value of x gets printed i.e. 23 is printed.

Scan for Video solution

12. (c)

```
if (b % 2 = = 0)
{ a = a*a; b= b/2; }// when b is even
else
{
res = res *a;// executes when b is odd
b = b –1;}
```

Let X = 3, Y = 5

This code will compute = 3^5

Initially, res = 1, a = 3, b = 5

1st iteration

As b is odd,

res = 1*3 = 3

b = 5 – 1 = 4

After 1st iteration

a = 3, b = 4, res = 3, X = 3, Y = 5

Let us check all option after

1st iteration (before 2nd iteration)

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

$3^5 = 3^4$(false)

(b) $(res*a)^Y = (res*X)^b$

$(3*3)^5 = (3*3)^4$(false)

(c) $X^Y = res*a^b$

$3^5 = 3*3^4$(true)

(d) $X^Y = (res*a)^b$

$3^5 = (3*3)^4$(false)

Hence option C is correct option (this is option elimination method).

Scan for Video solution

13. (c)

```
for i = 1 to n do
    for j = 1 to n do
        for k = j + 1 to n do
            D = D* 3
            1 multiplication
```

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=j+1}^{n} 1$$

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\left(\sum_{k=j+1}^{n} 1\right)$$

$$\sum_{k=j+1}^{n} 1$$

= 1 + 1 + 1 + ……..+1 (How many times ?)

k = j + 1 ……..k = n

= last – first + 1 = n– (j + 1) + 1

= n – j –1 + 1 = n –j

$$\boxed{\sum_{k=j+1}^{n} = (n-j)}$$

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n}(n-j)\right)$$

$(n-i) + (n-(i+1)) + (n-(i+2)) + \ldots (n-(n-1) + (n-n)$

$= (n-i) + (n-(i+1)) + (n-(i+2)) + \ldots + 1 + 0$

$= 0 + 1 + 2 + \ldots (n-i)$

$= 1 + 2 + \ldots\ldots (n-i)$

$$= \frac{(n-i)(n-i+1)}{2}$$

$$= \sum_{i=1}^{n}\frac{(n-i)(n-i+1)}{2}$$

$$= \sum_{i=1}^{n}\left[\frac{n^2+i^2+2in+n-1}{2}\right]$$

$$= \frac{1}{2}\sum_{i=1}^{n}n^2 + \frac{1}{2}\sum_{i=1}^{n}i^2 - \sum_{i=1}^{n}i.n + \frac{1}{2}\sum_{i=1}^{n}n - \frac{1}{2}\sum_{i=1}^{n}i$$

$$= \frac{n^3}{2} + \frac{n(n+1)(2n+1)}{12} - \frac{n.n(n+1)}{2} + \frac{n^2}{2} - \frac{n(n+1)}{2}$$

$$= \frac{n^3}{2} + \frac{n(n+1)(2n+1)}{12} - \frac{n^3}{2} - \frac{n^2}{2} + \frac{n^2}{2} - \frac{n^2}{4} - \frac{n}{4}$$

$$= \frac{n(n+1)(2n+1)}{12} - \frac{n(n+1)}{4}$$

$$= \frac{n(n+1)}{12}[2n+1-3]$$

$$= \frac{n(n+1)}{12}(2n-2)$$

$$= \frac{2\cdot n(n+1)(n-1)}{12}$$

$$= \frac{(n-1)(n)(n+1)}{6}$$

Therefor option (c) is correct answer.

Scan for Video solution

❑❑❑

CHAPTER

3

FUNCTIONS AND STORAGE CLASSES

Functions

1. [NAT] [GATE-2019 : 1M]

Consider the following C program:

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

The value printed by the program is _____.

2. [MCQ] [GATE-2019 : 2M]

Consider the following C program:

```
#include<stdio.h>
int r ( ){
    static int num = 7;
    return num – –;
}
int main( ){
    for (r ( ); r( ); r( ))
      printf ("%d", r( ));
    return 0;
}
```

Which one of the following values will be displayed on execution of the programs?

(a) 41 (b) 52

(c) 63 (d) 630

3. [NAT] [GATE-2018 : 2M]

Consider the following program written in pseudo-code. Assume that x and y are integers:

```
Count (x, y) {
if (y != 1) {
   if (x != 1) {
   print ("*");
         Count (x/2, y);
}
   else {
      y = y – 1;
          Count (1024, y);
    }
  }
}
```

The number of times that the print statement is executed by the call Count (1024, 1024) is _____.

Storage Classes

4. [NAT] [GATE-2015 : 1M]

The output of the following C program is_____

```
void f1 (int a, int b) {
int c;
c = a; a = b; b  =c;
}
    void f2 (int *a, int *b) {
     int c;
     c = *a; *a = *b; *b = c;
  }
     int main()   {
     int a = 4;b = 5,c = 6;
     f1 (a,b);
     f2 (&b,&c);
     printf ("%d",c-a-b);
}
```

5. [MCQ] [GATE-2014 : 2M]

Consider the C function given below.

```
int f( int j){
static int i = 50 ;
int k ;
if (i==j){
printf ("something");
k=f(i) ;
return 0
} else return 0 ;
```

Which one of the following is TRUE?

(a) The function returns 0 for all values of j.
(b) The function prints the string something for all values of j.
(c) The function returns 0 when j = 50.
(d) The function will exhaust the runtime stack or run into an infinite loop when j = 50.

Recursion

6. [NAT] [GATE-2023 : 1M]

The integer value printed by the ANSI-C program given below is _____ .

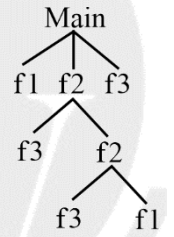
```
#include<stdio.h>
int funcp(){
static int x = 1;
x++;
return x;
}
int main(){
int x,y;
x = funcp();
y = funcp()+x;
printf("%d\n", (x+y));
return 0;
```

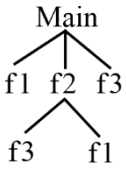
7. [MCQ] [GATE-2023 : 2M]

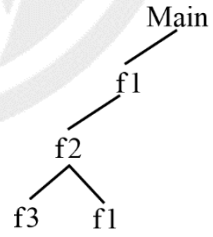
Consider the following program:

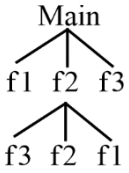
```
int main()
{
   f1();
   f2(2);
   f3();
   return(0);
}
int f1()
{
    return(1);
}
int f2(int X)
{
    f3();
  if (X==1)
     return f1();
  else
     return (X*f2(X-1));
}
int f3()
{
  return(5);
}
```

Which one of the following options represents the activation tree corresponding to the main function?

(a)


(b)


(c)


(d)


8. [NAT] [GATE-2021 : 1M]

Consider the following ANSI C function:

```
 int SomeFunction(int x, int y )
{
  if ((x==1) || (y==1)) return 1;
 if (x==y) return x;
if (x >y) return SomeFunction (x-y, y);
if (y >x) return SomeFunction (x, y-x);
}
```

The value returned by SomeFunction(15, 255) is____.

9. [NAT] [GATE-2021 : 2M]

Consider the following ANSI C program

```
#include < stdio.h >
int foo(int x, int y, int q) {
if ((x ≤ 0) &&(y ≤ 0))
return q;
if (x ≤ 0)
return foo (x, y–q, q);
if (y ≤ 0)
return foo (x–q, y, q);
return foo (x, y–q, q) + foo (x–q, y, q);
}
int main(){
   int r = foo (15, 15, 10);
   printf(" %d", r);
   return 0;
}
```

The output of the program upon execution is _____.

10. [MCQ] [GATE-2019 : 2M]

Consider the following C function.

```
void convert (int n){
   if (n < 0)
      printf("%d", n);
  else {
      convert(n/2);
      printf("%d", n % 2);
  }
}
```

Which one of the following will happen when the function convert is called with any positive integer n as argument?

(a) It will print the binary representation of n and terminate.

(b) It will print the binary representation of n in the reverse order and terminate

(c) It will print the binary representation of n but will not terminate

(d) It will not print anything and will not terminate.

11. [NAT] [GATE-2018 : 1M]

Consider the following C program:

```
#include<stdio.h>
int counter = 0;
int calc(int a, int b) {
    int c;
    counter++;
    if (b = = 3) return (a*a*a);
    else {
    c = calc(a, b/3);
    return(c*c*c);
    }
}
int main ( ) {
    calc(4, 81);
    printf ("%d", counter);
}
```

The output of this program is _____.

12. [MCQ] [GATE-2017 : 2M]

Consider the following two functions.

```
void fun1(int n) {
   if(n == 0) return;
   printf("%d", n);
   fun2(n – 2);
   printf("%d", n);
}
void fun2(int n) {
   if(n == 0) return;
   printf("%d", n);
   fun1(++n);
   printf("%d", n);
}
```

The output printed when fun1(5) is called is

(a) 53423122233445

(b) 53423120112233

(c) 53423122132435

(d) 53423120213243

13. [MCQ] [GATE-2017 : 2M]

Consider the C functions foo and bar given below:

```
int foo (int val)   {
int x=0;
while (val > 0)   {
        x = x + foo(val – –);
       }
 return val;
}
int bar (int val) {
int x = 0 ;
while (val >0){
     x = x + bar(val –1) ;
    }
return val;
}
```

Invocation of foo(3) and bar(3) will result in:

(a) Return of 6 and 6 respectively.
(b) Infinite loop and abnormal termination respectively
(c) Abnormal termination and infinite loop respectively
(d) Both terminating abnormally.

14. [MCQ] [GATE-2016 : 2M]

What will be the output of the following C program?

```
void count (int n ) {
   static int d=1;
   printf ("%d", n);
   printf ("%d", d);
   d++;
   if (n>1) count (n – 1);
   printf ("%d", d);
}
void main(){
   count (3);
}
```

(a) 312213444
(b) 312111222
(c) 3122134
(d) 3121112

15. [MCQ] [GATE-2016 : 2M]

Consider the recursive C function that takes two arguments unsigned int foo(unsigned int n, unsigned int r) {

```
if (n  > 0) return (n%r +  foo (n/r, r ));
else return 0;
}
```

What is the return value of the function foo when it is called as foo(513, 2)?

(a) 9 (b) 8
(c) 5 (d) 2

16. [MCQ] [GATE-2015 : 2M]

Consider the following recursive C function.

```
void get (int n) {
if (n < 1) return;
get (n – 1);
get (n – 3);
print f ("%d", n);
}
```

If get (6) function is being called in main( ) then how many times will the get( ) function be invoked before returning to the main()?

(a) 15 (b) 25
(c) 35 (d) 45

17. [MCQ] [GATE-2015 : 1M]

Consider the following function written in the C programming language.

```
void foo ( char * a){
if (* a&& * a !=' ')   {
foo (a+1)
putchar (* a)
}
}
```

The output of the above function on input "ABCD EFGH" is

(a) ABCD EFGH
(b) ABCD
(c) HGFE DCBA
(d) DCBA

18. [NAT] [GATE-2015 : 1M]

Consider the following C function.

```
int fun (int n){
int x=1, k ;
if (n==1) return x ;
for (k=1 ; k<n ;++k)
x=x+ fun (k) *  fun (n-k) ;
return x ;
```

The return value of fun (5) is

19. [NAT] [GATE-2014 : 1M]

Consider the function func shown below:

```
int func (int num)
{    int count =0 ;
   while (num)
  { count ++;
    num >>=1 ;
  }
   return (count);
}
```

The value returned by function (435) is_____.

20. [MCQ] [GATE-2012 : 2M]

Consider the following C code segment.

```
int a, b, c = 0;
void prtFun(void);
main()
{
    static int a = 1;       /* Line 1 */
    prtFun();
    a += 1;
    prtFun();
     printf(" \n %d %d ", a, b);
}
void prtFun(void)
{
    static int a = 2;       /* Line 2 */
     int b = 1;
     a += ++b;
     printf(" \n %d %d ", a, b);
}
```

What output will be generated by the given code segment?

(a) 3 1
4 1
4 2

(b) 4 2
6 1
6 1

(c) 4 2
6 2
2 0

(d) 4 2
4 2
2 0

21. [MCQ] [GATE-2012 : 2M]

Consider the following C code segment

```
int a, b, c = 0;
void prtFun(void);
main()
{
    static int a = 1;       /* Line 1 */
    prtFun();
    a += 1;
    prtFun();
     printf(" \n %d %d ", a, b);
}
void prtFun(void)
{
    static int a = 2;       /* Line 2 */
     int b = 1;
     a += ++b;
     printf(" \n %d %d ", a, b);
}
```

What output will be generated by the given code segment if:

Line 1 is replaced by auto int a = 1;

Line 2 is replaced by register int a = 2;

What output will be generated by the given code segment?

(a) 3 1
4 1
4 2

(b) 3 1
5 2
5 2

(c) 4 2
6 2
2 0

(d) 4 2
4 2
2 0

22. [MCQ] [GATE-2011 : 2M]

Consider the following recursive C function that takes two arguments. unsigned int foo (unsigned int n, unsigned int r)

```
{
 if (n>0) return ((n%r)+ foo(n/r,r));
 else return 0;
}
```

What is the return value of the function foo when it is called as foo (345,10) ?

(a) 345 (b) 12

(c) 5 (d) 3

23. [MCQ] [GATE-2009 : 1M]

Consider the program below:

```
#include<stdio.h>
int fun(int n, int*f_p) {
int t,f;
if(n<=1) {
 *f_p=1;
    return 1;
}
t=fun(n-1,f_p);
f=t+*f_p;
*f_p=t;
  return f;
}
int main( )
{
   int x=15;
   printf("%d\n",fun(5,&x));
   return 0;
}
```

The value printed is

(a) 6

(b) 8

(c) 14

(d) 15

Scoping

24. [NAT] [GATE-2020 : 2M]

Consider the following C functions:

```
int fun1(int n){
static int i = 0
if (n > 0) {
        ++i;
        fun1(n-1);
       }
return (i);
}
int fun2 (int n) {
static int i = 0;
if (n > 0) {
      i = i + fun1(n);
      fun2(n–1);
      }
return (i);
}
```

The return value of fun2 (5) is _____.

25. [NAT] [GATE-2015 : 2M]

Consider the following C program.

```
#include<stdio.in>
int  f1(void);
int  f2(void);
int  f3(void);
int  x=10;
int main()
{
   int x=1;
    x+=f1()+f2()+f3()+f2();
   print f("%d", x);
   return 0;
   }
   int f1(){int  x=25;x++;return x;}
   int f2(){static int=50;x++;return x;}
   int f3(){x*=10;return x;}
```

The output of the program is _____

❑❑❑

ANSWER KEY

1.	(26 to 26)	**2.**	(b)	**3.**	(10230 to 10230)	**4.**	(–5 to –5)
5.	(d)	**6.**	(7 to 7)	**7.**	(a)	**8.**	(15 to 15)
9.	(60 to 60)	**10.**	(d)	**11.**	(4 to 4)	**12.**	(a)
13.	(c)	**14.**	(a)	**15.**	(d)	**16.**	(b)
17.	(d)	**18.**	(51 to 51)	**19.**	(9 to 9)	**20.**	(c)
21.	(d)	**22.**	(b)	**23.**	(b)	**24.**	(55 to 55)
25.	(230 to 230)						

SOLUTIONS

1. (26 to 26)

$x_{main} = 2$ 26

$y_{main} = 5$ 12

1) y_{main} = jumble(5,2)

x=12, y=2

x=2*x+y

x=2*5+2

x=12

after 1st jumble function call

x_{main}=2, y_{main}=12

2) x_{main}=jumble(12,2)

x=26,y=2

x=2*x+y

x=2*12+2

x=26

after 2nd jumble function call

x_{main}=26, and y_{main}=12

lastly value of x is printed which is 26.

Scan for Video solution

2. (b)

r(): it will return the current value of num and returns num=num-1(post decrement).

```
                (5) (8)
      (1)     (2)    (4) (7)
for   (r ()) ;  r();    r( ))
                {
              (3) printf ("% d ", r ())
               (6)
                {
```

Post-decrement: A post-decrement operator is used to decrement the value of a variable after executing the expression in which the operator is used.

(1) r() will return 7 and then num is decreased by 1 that is num =7, value 7 is printed and then decremented.

(2) r() will return 6(which is non zero) that's why condition will become true and code inside the loop will be executed, but before reaching printf statement num is decresed by 1 ie…. num becomes 5.

(3) printf("%d"r());
r() inside printf will return 5 and same is printed and after printing num becomes 4.

(4) r() is called and 4 is returned and then decreses the num value by 1, hence num becomes 3.

(5) r() will return 3 which is non zero(true) so printf will be executed, before printf num value becomes 2 because of post decrement.

(6) printf("%d",r()), r() returns 2 and same value is printed. And then num becomes 1.

(7) r(), will return 1 and decrements the value to zero.

(8) r(), Here zero is returned and condition of for loop becomes false hence loop gets terminated.

Therefore, output printed is 52 which is option b.

Scan for Video solution

3. (10230 to 10230)

* If y =1, then the function will terminate without printing anything.

Count (x, 1) will not print anything

else statement will be executed when if condition fails.

Count (1024, 1024)

↓ (1) print

Count (512, 1024)

↓ (2) print

Count (256, 1024)

↓ (3) print

Count (128, 1024)

↓ (4) print

Count (64, 1024)

↓ (5) print

Count (32, 1024)

↓ (6) print

Count (16, 1024)

↓ (7) print

Count (8, 1024)

↓ (8) print

Count (4, 1024)

↓ (9) print

Count (2, 1024)

↓ (10) print

Count (1, 1024)

Note:- for each such y, 10 times printf will take place

Count (1024, 1024)

↓After 10 times printf

Count (1, 1024)

Because x = 1 else part will be executed

Count (1024, y) $y \neq 1$

↓10 times print

Count (1, y)

$\Rightarrow$ x = 1 else part will be executed

1. Count (1024, 1024)

↓ after 10 time printing

Count (1, 1024)

$y = y - 1 \Rightarrow y = 1023$

2. Count (1024, 1023) will execute

Count (1024, 1023)

↓ 10 time

Count (1, 1023)

Again, else part will execute

y = 1022

count (1024, 1022) will execute

count (1024, 1024) $\rightarrow$ 10 times

count (1024, 1023) $\rightarrow$ 10 times

count (1024, 1022) $\rightarrow$ 10 times

.

.

.

Count (1024, 2) $\rightarrow$ 10 times

i.e. count (1024, y)where y = 2, 3, …1024

i.e. for 1023 value of y count (1024, y) will print 10 times

Total number of printf = $10 \times 1023 = 10230$

Scan for Video solution

4. (–5 to –5)

f2() function is swapping.

⇒ As f2 is called by passing address of b and address of c.

f2 () ⇒ will swap the contents of b, c.

a[4 b[6 c[5

printf will point the value of c – a – b

5 – 4 – 6

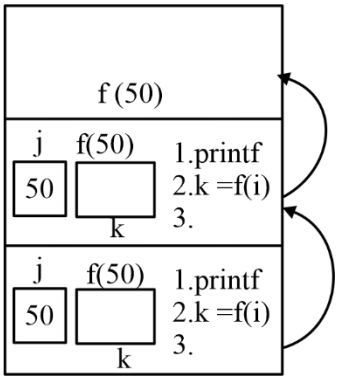
= 5 – 10

= – 5

Scan for Video solution

5. (d)

```
int f(int j){
static int i=50;
int k;
if(i= =j){
1.printf("something");
2.k=f(i);
3.return 0;
}
else
return 0
}
```



f(50) will call f(50) and keeps on calling which will lead to stack overflow.

for j = 50, function will never reach return statement.

Option (a): is wrong because for j=50, stack overflow occurs and function never reaches return statement.

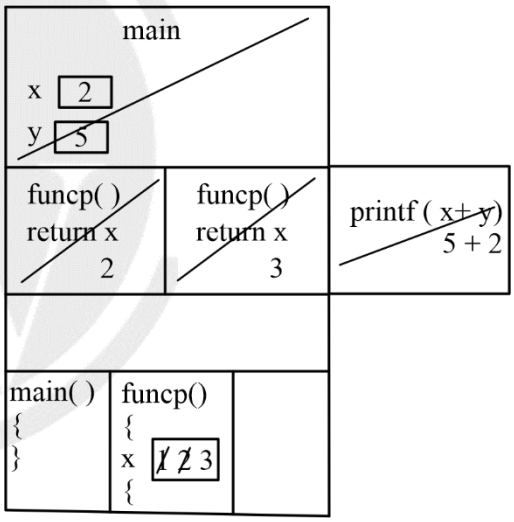
Option (c): when j=50, function will never reach return statement.

Option(b): for j!=50, function will not print anything that's why (b) is also wrong.

Option(d): as explained through diagram which shows that function will exhaust the runtime stack or run into an infinite loop when j=50, therefore (d) is correct answer.

Scan for Video solution

6. (7 to 7)



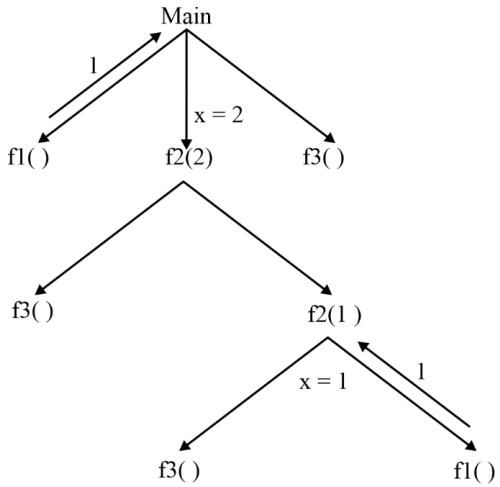
After execution of funcp 2 times, printf function prints the output (7) and gets deleted from activation record.

Similarly, control is returned to the main function, main function also gets deleted from activation record and control is returned to operating system.

Hence 7 is printed.

Scan for Video solution

7. (a)



Comparing the above tree with the given trees in options. Above tree matches with the tree given in option A, hence a is correct answer.

Scan for Video solution

8. (15 to 15)

SomeFunction (15, 255)
15 ↺ y > x

SomeFunction (15, 240)
15 ↺ y > x

SomeFunction (15, 225)
15 ↺ y > x

SomeFunction (15, 210) → 210 is a multiple of 15 (15* 14 = 210)
15 ↺ y > x

SomeFunction (15, 30)
15 ↺ y > x

Note: The above code is finding the GCD of 2 number using repeated subtraction.

SomeFunction (15, 15)
15 ↺ x = = y

return 15

∴ Correct answer is 15

Scan for Video solution

9. (60 to 60)

Lets first understand the fragments of code.

1) if((x ≤ 0)&&(y ≤ 0))

The above case states that when both x and y are less than 0 (or) equal to zero then return 3rd argument(q).

2) if(x ≤ 0)

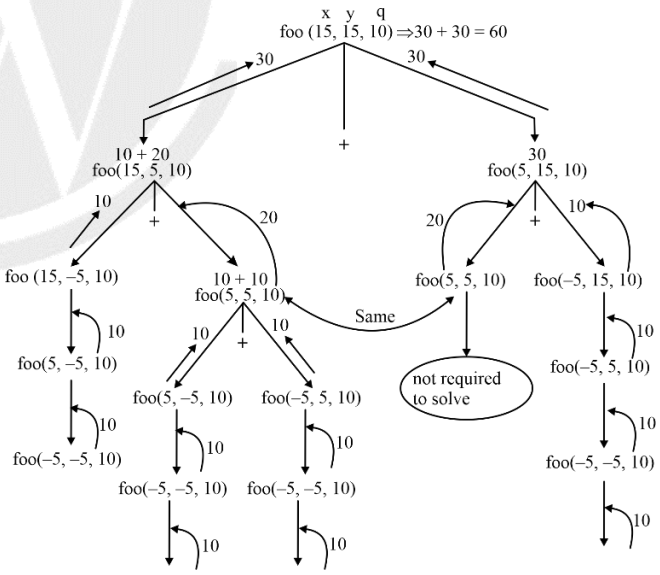
The above case states that x ≤ 0 and y > 0 then decrement 2nd argument using 3rd argument.

3) if(y ≤ 0)

The above case states that x >0, y ≤ 0 then decrement first argument using 3rd argument.

4) return foo (x, y – q, q) + foo (x – q, y, a)

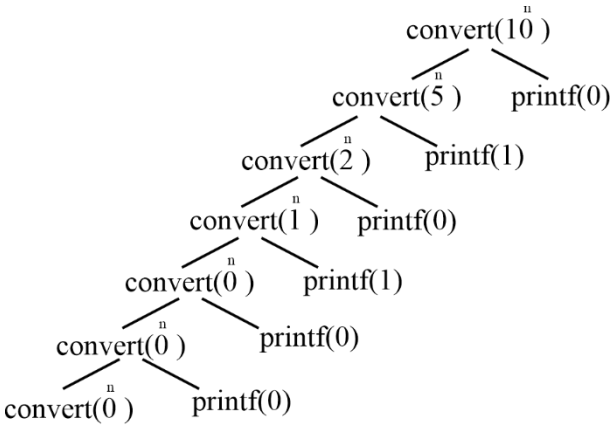
The above case states that, when both x, y > 0 then from first function call, decrease 2nd argument using 3rd argument and from 2nd function call, decrease first argument using 3rd argument.



Scan for Video solution

10. (d)

Let us try for n = 10



- Every printf is waiting, statement written after recursive call executes in opposite order of call.
- Nothing will be printed because convert(0) keeps on calling convert(0) and it will never terminate.

Scan for Video solution

11. (4 to 4)

In this program we need not focus on the return value, we must focus on the counter variable. We need to just look for the number of times calc function is being called.

```
      a   b
calc (4, 81)
      |  counter = 1
      ↓      b = 81
calc (4, 27)
      |  counter = 2
      ↓      b =27
calc (4, 9)
      |  counter = 3
      ↓      b =9
calc (4, 3)
      |  counter = 4
      ↓      b =3
```

No further call

∴ value of counter variable is printed is 4

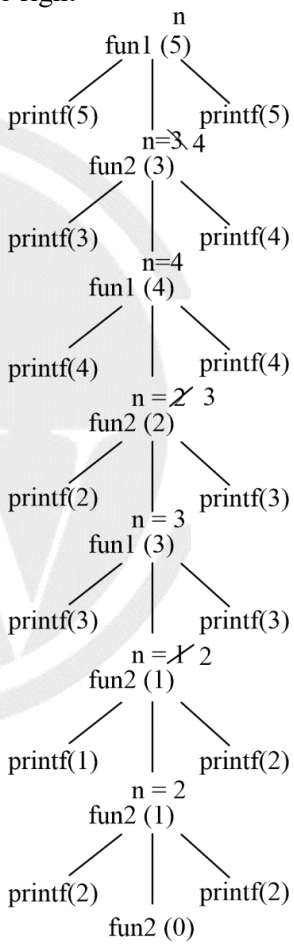
Scan for Video solution

12. (a)

$$\text{fun1}(++n);\begin{cases} n = n+1 \\ \text{fun1}(n) \end{cases}$$

printf ("% d", n) → will print incremented value of n

We need to trace this tree from top to bottom & left to right



We need to trace this tree from top to bottom & left to right.

Hence, 53423122233445 is printed, therefore option (a) is correct

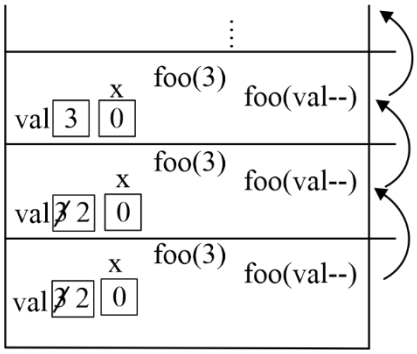
Scan for Video solution

13. (c)

Understanding foo function

x=x+foo(val--) ;

val--performs post decrement.foo(val) will be called and then val=val-1 is performed later(after function call)

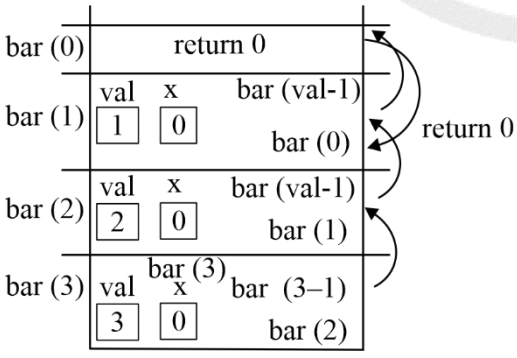


foo(3) will call foo(3) which will again call foo(3) and gets called every time.

Infinite recursive call keeps on increasing stack size, but stack size is limited. Hence at some point of time stack overflow occurs ie…. abnormal termination takes place.

Understanding bar function:

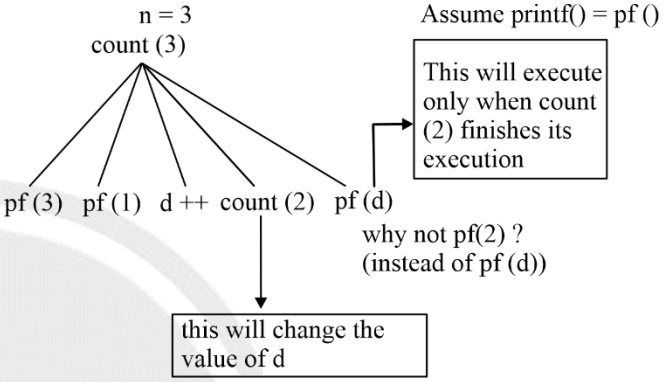
```
int bar(int val){
int x=0;
while(val>0){
x=x+bar(val-1);}
return val;
}
```

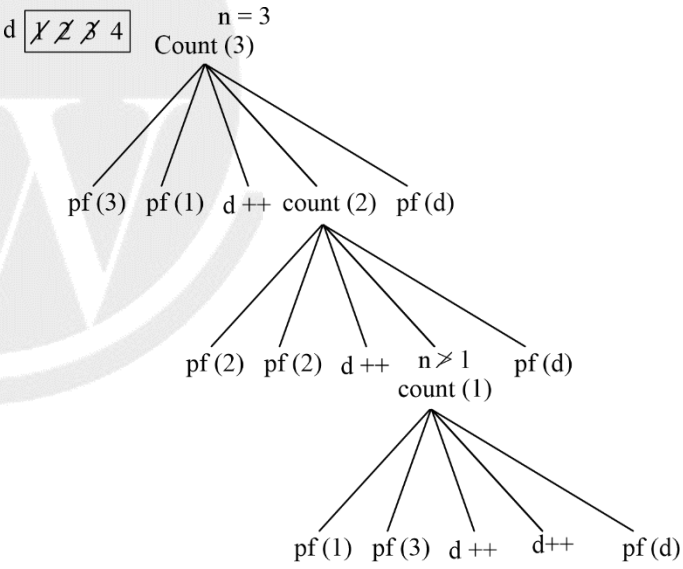


- We are still inside loop and val > 0 is true, bar (1) will again call bar(0) and again bar(0) will return 0,
- This process will continue .. loop count wont be getting terminated.
- But, stack size is not being exceeded at any moment of time.
- Leading to infinite loop Hence, correct option is (c).

Scan for Video solution

14. (a)





On tracing the tree from top to bottom and left to right we get

3,1,2,2,1,3,4,4,4

∴ Option a is correct answer.

Scan for Video solution

15. (d)

2

n r

foo (513, 2)

1

n r

1+ foo (256, 2)

1

n r

0+ foo (128, 2)

1

n r

0+ foo (64, 2)

1

n r

0+ foo (32, 2)

1

n r

0+ foo (16, 2)

1

n r

0+ foo (8, 2)

1

n r

0+ foo (4, 2)

1

n r

0+ foo (2, 2)

1

0+ foo (1, 2)

0

1+ foo (0, 2)

0

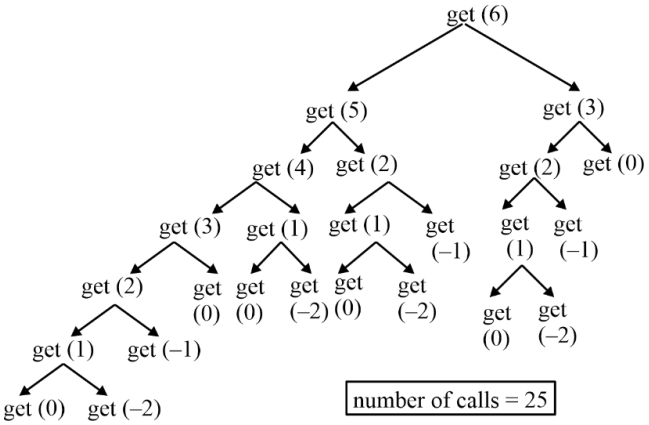
Answer is option (d)

Scan for Video solution

16. (b)

```
void get (int n) {
if (n < 1) return;
get (n – 1); ⇒ Recursion call with 1 less argument
get (n – 3); ⇒ Recursion call with 3 less argument
printf("%d", n);
```

Need not to focus on printf statement



number of calls = 25

Scan for Video solution

17. (d)

&&! =

Priority of && is less than !=

if ((*a)) && (*a! = ' '))

if will become false, Either

(1) *a = 0 (ASCI to or \0)

(2) *a become ' '

Now,

Recursion 'A'→'B'→'C'→'D'→ Recursion ends and then it starts printing.

Printing is done because putchar (*a); is written after recursive call.

Hence, DCBA is printed, i.e. option D is correct.

Scan for Video solution

18. (51 to 51)

x is a local variable so in every call x →1

fun(1) =1

fun(2) ⇒ = 1

x = x + fun(1)*fun (2 –1)

= 1 + fun (1) * fun(1) = 1 + 1 × 1 = 2

fun(3) ⇒ k = 1, 2

k = 1

x = x + fun(1) * fun (2) + fun(2) *fun(1)

= 1 + 1 × 2 + 2 × 1 = 1 + 2 + 2 = 5

fun (4) k = 1, 2, 3 n =4

x = x + fun(1) × fun(3) + fun(2) × fun(2) + fun(3) × fun(1)

= 1 + 1 × 5 + 2 ×2 + 5 × 1 = 15

k = 1, 2, 3, 4 n = 5

fun(5)

= 1 + fun(1) × fun(4) + fun(2) × fun(3) + fun(3) × fun(2) + fun(4) × fun(1)

= 1 + 1 × 15 + 2 × 5 + 5 × 2 + 15 × 1

= 1 + 15 + 10 + 10 + 15

= 51

Scan for Video solution

19. (9 to 9)

num = 435		count ++ (✓)
num >> =1	num = 217	count ++ (✓)
num >> =1	num = 108	count ++ (✓)
num >> =1	num = 54	count ++ (✓)
num >> =1	num = 27	count ++ (✓)
num >> =1	num = 13	count ++ (✓)
num >> =1	num = 6	count ++ (✓)
num >> =1	num = 3	count ++ (✓)
num >> =1	num = 1	count ++ (✓)

} 9 times

num>>1 (divided by 2) → num = 0

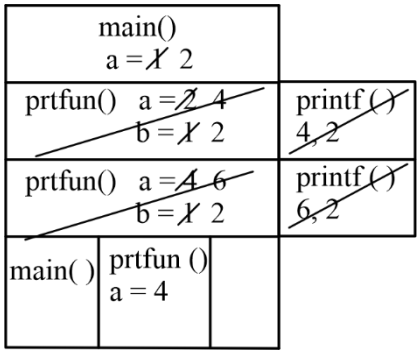
Hence, 9 is the correct answer.

Scan for Video solution

20. (c)

Part-A

a = 0, b = 0, c = 0



- 4, 2 is printed and it is deleted from activation record.
- Again prtfun () is called
- 6,2 is printed and it is deleted from activation Record, now control is retunred to main and 2,0 is printed.

4 2
6 2 } is pointed, Hence (c) is correct option
2 0

Scan for Video solution

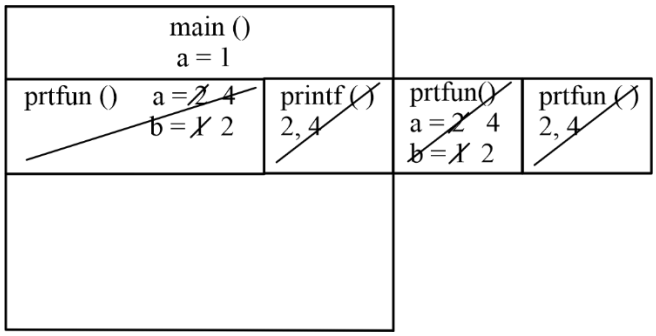
21. (d)

Part -B

a = 0,

b = 0

c = 0



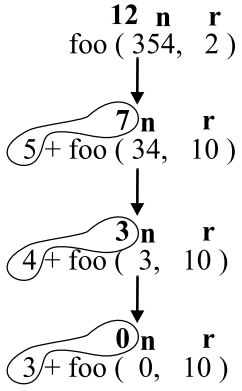
Note that register vairable is as same as local vairable except the storage area

- After printing 4, 2 printf function and printf( ) gets deleted from activation record and control is returned to main, again prtfun is called.
- Again 4, 2 is printed and both the functions get deleted from activation record and control is returned to main and 2,0 is printed.

$\therefore \begin{matrix} 4\ 2 \\ 4\ 2 \\ 2\ 0 \end{matrix}$ is pointed Hence, option (d) is correct answer

Scan for Video solution

22. (b)

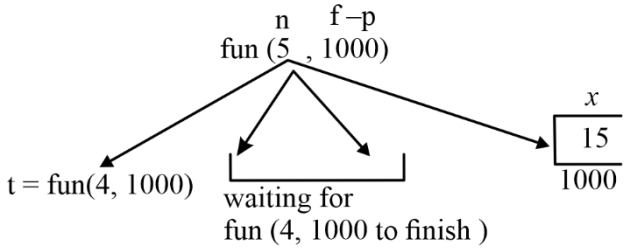


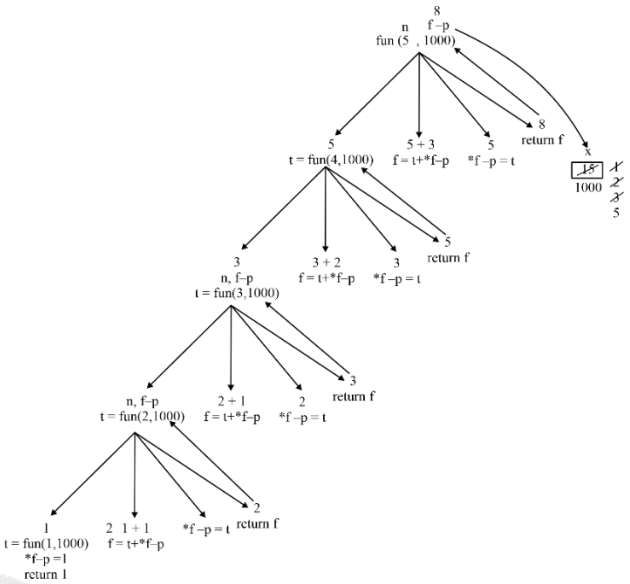
Hence (b) is correct option

Scan for Video solution

23. (b)

x = 15





Correct answer is 8 i.e. option (b).

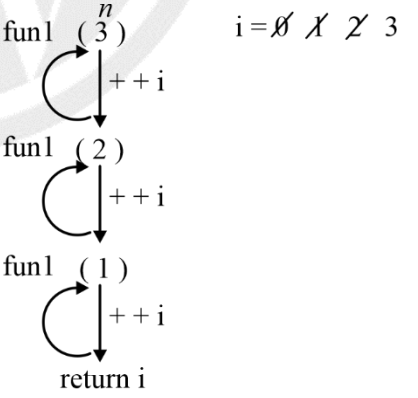
Scan for Video solution

24. (55 to 55)

Let us try to understand the working of function fun1

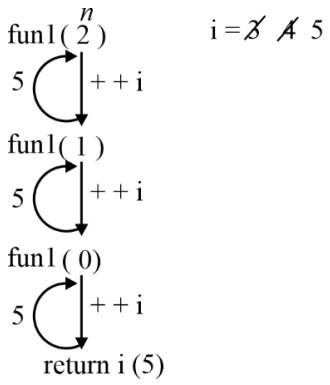
Let fun1(3) is called first

i is an static variable



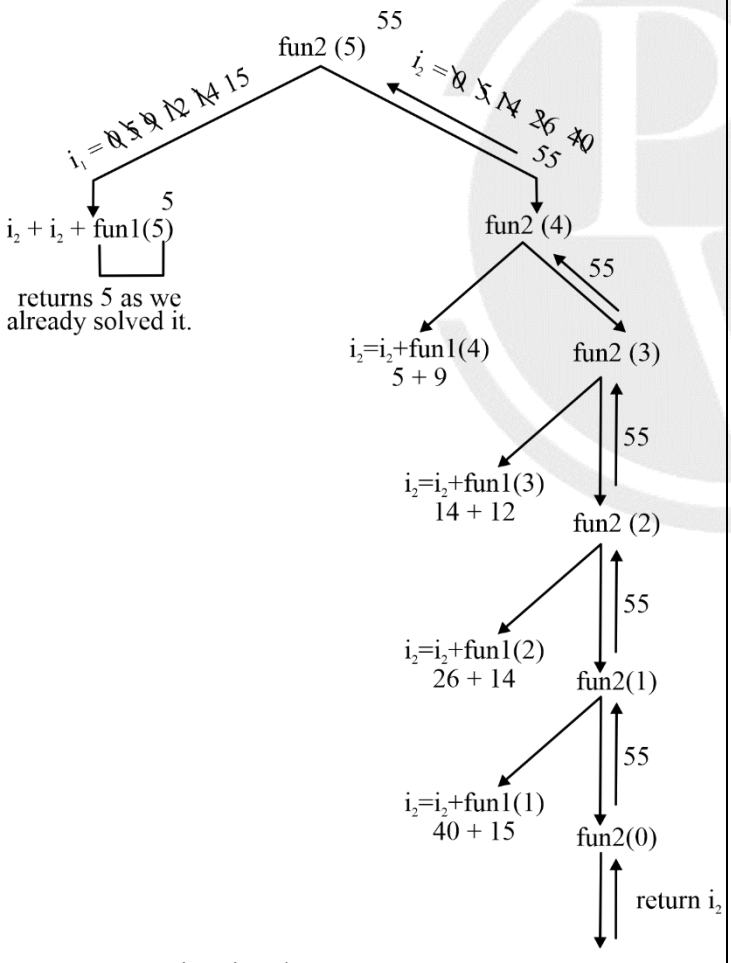
Before fun1(3) the value of i was 0 and after fun1(3) the value of i becomes 3.

Let us assume fun1() is called again i.e. for the 2nd time.



- Before fun1(2) value of static variable i was 3 and after fun1(2) value returned is 5. Which is the value of i variable
- fun 1(n) is increasing the value of static int i variable and returning the incremented value.

Now let us understand fun2(5).



∴ 55 is printed

Scan for Video solution

25. (230 to 230)

x_{main}	x_{global}
1	10

$x_{main} = x_{main} + f_1( ) + f_2() + f_3() + f_2();$

$f_1()$ is called:

x_1

~~25~~

26

26 is retuned by $f_1()$

$x_{main} = x_{main} + f_1( ) + f_2() + f_3() + f_2();$

$= 1 + 26$

$f_2()$ is called:

There is local static variable which resides in memory throughout the program.

x ~~50~~

51

51 is returned but x being static, still remains in memory.

$x_{main} = x_{main} + f_1( ) + f_2() + f_3() + f_2();$

$= 1 + 26 + 51$

$f_3()$ is called:

$x_{global} * = 10$

$x_{global} = 10 * 10$

$x_{global} = 100$

100 is returned

$x_{main} = x_{main} + f_1( ) + f_2() + f_3() + f_2();$

$= 1 + 26 + 51 + 100$

$f_2()$ is called again

Variable x is already residing in memory already.

x ~~51~~ 52

52 is returned

$x_{main} = x_{main} + f_1( ) + f_2() + f_3() + f_2();$

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

∴ 230 is printed

Scan for Video solution

❑❑❑

CHAPTER

4 POINTERS AND STRINGS

Arrays and Pointers

1. [MCQ] [GATE-2022 : 1M]

What is printed by the following ANSI C program?

```
#include<stdio.h>
int main(int argc, char *argv[])
{
int x = 1, z[2] = {10, 11};
int *p = NULL;
p = &x;
*p = 10;
p = &z[1];
*(&z[0] + 1) += 3;
printf("%d, %d, %d\n", x, z[0], z[1]);
return 0;
}
```

(a) 1, 10, 11 (b) 1, 10, 14

(c) 10, 14, 11 (d) 10, 10, 14

2. [MCQ] [GATE-2022 : 2M]

What is printed by the following ANSI C program?

```
#include<stdio.h>
int main(int argc, char *argv[])
{
int a[3][3][3] =
{{1, 2, 3, 4, 5, 6, 7, 8, 9},
 {10, 11, 12, 13, 14, 15, 16, 17, 18},
 {19, 20, 21, 22, 23, 24, 25, 26, 27}};
    int i = 0, j = 0, k = 0;
    for( i = 0; i < 3; i++ ){
        for(k = 0; k < 3; k++ )
     printf("%d ", a[i][j][k]);
        printf("\n");
    }
        return 0;
}
```

(a) 1 2 3
10 11 12
19 20 21

(b) 1 4 7
10 13 16
19 22 25

(c) 1 2 3
4 5 6
7 8 9

(d) 1 2 3
13 14 15

3. [NAT] [GATE-2021: 2M]

Consider the following ANSI C function:

```
int SimpleFunction (int Y[ ], int n, int x)
{
int total =Y[0], loopIndex;
for ( loopIndex =1 ; loopIndex <=n-l; loopIndex ++)
total = x * total +Y[ loopIndex ] ;
return total;
}
```

Let Z be an array of 10 elements with Z[i]=1, for all i such that $0 \le i \le 9$.

The value returned by SimpleFunction (Z, 10,2) is____

4. [MCQ] [GATE-2021 : 1M]

Consider the following ANSI C program.

```
#include <stdio.h >
int main( ){
int arr[4][5];
int i, j;
for (i=0 ; i<4 ; i++){
for (j=0 ; j<5 ; j++){
arr [i][j]=10 * i + j;
```

```
}
}
printf(" % d ", *(arr[1]+9));
return 0;
}
```

What is the output of the above program?

(a) 14 (b) 20
(c) 24 (d) 30

5. [NAT] [GATE-2020 : 1M]

Consider the following C program:

```
#include < stdio.h >
int main ( ) {
   int a [4][5] = {{1, 2, 3, 4, 5},
                   {6, 7, 8, 9, 10},
                   {11, 12, 13, 14, 15},
                   {16, 17, 18, 19, 20}};
   printf("%d", *(*(a+**a+2)+3));
   return (0);
}
```

The output of the program is_____.

6. [NAT] [GATE-2020 : 2M]

Consider the following C functions:

```
int tob(int b, int* arr){
            int i;
     for (i = 0; b>0; i++) {
          if (b%2) arr [i] = 1;
          else        arr [i] = 0;
          b= b/2;
           }
  return (i);
}
int pp (int a, int b){
              int arr [20];
    int i, tot = 1, ex, len;
     ex = a;
    len = tob (b, arr);
    for (i = 0; i< len; i++){
                if (arr[i] ==1)
                 tot = tot * ex;
               ex = ex * ex;
                }
            return (tot);
    }
```

The value returned by pp(3, 4) is_____.

7. [NAT] [GATE-2019 : 1M]

Consider the following C program:

```
#include <stdio.h>
int main ()
{
 int arr [ ] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 5}, *ip =
arr + 4;
 printf ("%d \n", ip[1]);
 return 0;
}
```

The number that will be displayed on execution of the program is _____.

8. [NAT] [GATE-2019 : 2M]

Consider the following C program:

```
#include<stdio.h>
int main( )
{
   int a[ ] = {2, 4, 6, 8, 10};
   int i, sum = 0, *b = a + 4;
   for (i = 0; i < 5; i++)
   sum = sum + (*b – i) –*(b – i);
   printf("%d\n", sum);
   return 0;
}
```

The output of the above C program is_____.

9. [MCQ] [GATE-2018 : 1M]

Consider the following C program:

```
# include<stdio.h>
struct Ournode {
    char x, y, z;
} ;
int main () {
    struct Ournode p = {'1' , '0' ,  'a' + 2};
    struct Ournode *q = &p;
```

```
    printf ("%c , %c",*((char*)q + 1), *((char*)
    q + 2));
    return 0;
}
```

The output of this program is:

(a) 0, c (b) 0, a+2
(c) ' 0 ', ' a+2 ' (d) ' 0 ', ' c '

10. [MCQ] [GATE-2017 : 1M]

Consider the following C code:

```
#include<stdio.h>
int *assignval (int  *x , int val)
  { *x = val;
   return x;
   }
   void main ( )
{
            int *x = malloc (sizeof(int));
if (NULL == x) return;
x = assignval (x, 0);
if ( x )
{    x = (int *) malloc (sizeof(int));
 if (NULL = = x) return;
 x = assignval (x, 10);
 }
    printf ("%d\n", *x);
    free(x);
}
```

The code suffers from which one of the following problems:

(a) compiler error as the return of malloc is not typecast appropriately.
(b) compiler error because the comparison should be made as x== NULL and not as shown.
(c) compiles successfully but execution may result in dangling pointer
(d) compiles successfully but execution may result in memory leak.

11. [NAT] [GATE-2017 : 2M]

Consider the following snippet of a C program. Assume that swap (&x, &y) exchanges the content of x and y.

```
int main ( )  {
int array [ ] = {3 , 5 , 1 , 4 , 6 , 2};
int done = 0;
int i;
while ( done ==0){
       done =1
       for (i=0 ; i<=4 ; i++){
            if (array [i] < array [i+1]){
                swap(&array[i], &array[ i  + 1]);
               done =0;
              }
        }
}
for (i=5 ; i>=1 ; i– –)
     {
       if (array [i] > array [i-1])
        {
         swap (&array[i], &array[i–1]);
        done =0;
         }
      }
  }
printf ("%d", array [3]);
}
```

The output of the program is_____

12. [NAT] [GATE-2017 : 2M]

Consider the following C program.

```
#include<stdio.h>
#include<string.h>
int main( )
{
   char* c= "GATECSIT2017";
   char* p=c ;
   printf("%d ", (int)strlen(c+2[p]–6[p]–1)) ;
   return 0 ;
}
```

The output of the program is_____.

13. [MCQ] [GATE-2017 : 1M]

Consider the following function implemented in C :

```
void printxy(int  x , int  y )
{
int  *ptr;
x = 0;
ptr = &x;
y = * ptr;
*ptr = 1;
printf("%d, %d ", x, y) ;
}
```

The output of invoking printxy(1, 1) is:

(a) 0, 0 (b) 0, 1

(c) 1, 0 (d) 1, 1

14. [MCQ] [GATE-2017 : 1M]

Match the following:

List-I	List-II
P. static char var;	i. Sequence of memory locations to store addresses
Q. m=malloc(10) ; m= NULL;	ii. A variable located in data section of memory
R. char *ptr [10];	iii. Request to allocate a CPU register to store data
S. register int var1;	iv. A lost memory which cannot be freed

(a) P → (ii), Q → (iv), R → (i), S → (iii)

(b) P → (ii), Q → (i), R → (iv), S → (iii)

(c) P → (ii), Q → (iv), R → (iii), S → (i)

(d) P → (iii), Q → (iv), R → (i), S → (ii)

15. [NAT] [GATE-2016 : 2M]

Consider the following program:

```
int  f(int *p, int n){
   if (n <= 1) return 0;
   else return max(f(p+1, n – 1),p[0] – p[1]);
}
int main(){
   int a[ ] = {3 , 5 , 2 , 6 , 4};
   printf("%d", f(a, 5));
}
```

Note: max (x, y) returns the maximum of x and y:

The value printed by this program is_____.

16. [NAT] [GATE-2016 : 1M]

The value printed by the following program is____.

```
void f(int  * p,  int m) {
      m = m + 5;
     *p = *p + m;
      return;
}
void main( ) {
    int i = 5, j = 10;
    f( &i, j);
    printf("%d"  ,i + j);
}
```

17. [MCQ] [GATE-2016 : 2M]

What will be the output of the following pseudo-code when parameters are passed by reference and dynamic scoping is assumed?

```
a = 3;
void n(x) {
x = x * a; print (x);
}
void m(y) {
a = 1 ; a = y – a;  n(a);
print (a); }
void main () {
m(a);
}
```

(a) 6, 2 (b) 6, 6

(c) 4, 2 (d) 4, 4

18. [NAT] [GATE-2016 : 1M]

Consider the following C program.

```
#include<stdio.h>
void mystery(int*ptra,int*ptrb)
{    int*temp;
     temp=ptrb;
     ptrb=ptra;
     ptra=temp;
}
```

```
    int main()
{   int a=2016, b=0, c=4, d=42;
    mystery(&a, &b);
    if(a<c)
mystery(&c, &a);
mystery(&a, &d);
printf("%d\n",a);
}
```

The output of the program is____.

19. [MCQ] [GATE-2016 : 1M]

Consider the following C program.

```
void f (int, short);
void main (){
int i=100;
short s=12;
short  * p=& s ;
_____ ; / /  call to f()
}
```

Which one of the following expressions, when placed in the blank above, will NOT result in a type checking error?

(a) f(s, * s) (b) i=f(i, s)
(c) f (i,* s) (d) f(i, * p)

20. [NAT] [GATE-2015 : 2M]

Consider the following C program.

```
# include < stdio.h >
int main () {
    static int a[ ] = {10, 20, 30, 40, 50};
    static int *p[ ] = {a,a + 3, a+4, a + 1, a + 2);
    int **ptr = p;     ptr++;
    printf ("%d % d", prt-p, **ptr);
}
```

The output of the program is _____.

21. [MCQ] [GATE-2015 : 1M]

Consider the following C program segment.

```
# include <stdio.h >
int main (){
char s1 [7]=" 1234 ", * p;
p = s1 + 2;
* p=' 0 ';
printf("% s" ,  s1);
}
```

What will be printed by the program?

(a) 12 (b) 120400
(c) 1204 (d) 1034

22. [MCQ] [GATE-2014 : 1M]

Consider the following program in C language:

```
#include <stdio.h>
main()
{
  int i;
  int*pi = &i;
  scanf("%d",pi);
  printf("%d\n", i+5);
}
```

Which one of the following statements is TRUE?

(a) Compilation fails.
(b) Execution results in a run-time error.
(c) On execution, the value printed is 5 more than the address of variable i.
(d) On execution, the value printed is 5 more than the integer value entered.

23. [MCQ] [GATE-2010 : 2M]

What is the value printed by the following C program?

```
#include < stdio.h>
int f(int *a, int n){
if (n <= 0) return 0;
else if (*a % 2 == 0)
return *a + f(a + 1, n-1);
else return *a - f(a + 1, n -1);
}
int main ()
{
int a[] = {12, 7,13.4.11,6);
printf(<,%d', f(a, 6));
return 0;
}
```

(a) –9 (b) 5
(c) 15 (d) 19

24. [MCQ] [GATE-2010 : 1M]

What does the following program print?

```
# include <studio.h>
void f( int * p, int * q){
p = q ;
* p = 2 ;
} int i = 0, j = 1 ;
int main( )
{
f(&i, &j) ;
printf(" % d% d\n ", i, j) ;
getchar(); return 0; }
```

(a) 22 (b) 21
(c) 01 (d) 02

Common Data for next two questions:

Consider the following C program that attempts to locate an element x in an array Y[] using binary search. The program is erroneous.

```
1.  f( int Y[10], int x){
2.  int i, j, k;
3.  i=0 ; j=9;
4.  do {
5.  k = (i + j) / 2;
6.  if (Y[K]<x) i = k; else j = k;
7.  } while ((Y[k]    !=x) &&(i<j));
8.  if (Y[k]==x)
      printf("x is in the array ") ;
9.  else
      printf ("x is not in the array");
10. }
```

25. [MCQ] [GATE-2008 : 2M]

On which of the following contents of Y and x does the program fail?

(a) Y is [1 2 3 4 5 6 7 8 9 10] and x < 10
(b) Y is [1 3 5 7 9 11 13 15 17 19] and x < 1
(c) Y is [2 2 2 2 2 2 2 2 2 2] and x > 2
(d) Y is [2 4 6 8 10 12 14 16 18 20] and 2 < x < 20 and x is even

26. [MCQ] [GATE-2008 : 2M]

The correction needed in the program to make it work properly is

(a) change line 6 to: if (Y[k] < x) i = k + 1; else j = k – 1;
(b) change line 6 to: if (Y[k] < x) i = k - 1; else j = k + 1;
(c) change line 6 to: if (Y[k] < x) i = k; else j = k;
(d) change line 7 to: while ((Y[k] = = x) && (i < j));

Strings

27. [MCQ] [GATE-2018 : 2M]

Consider the following C program:

```
#include<stdio.h>
void fun1 (char*  s1 , char* s2) {
   char* temp;
   temp = s1;
   s1 = s2;
   s2 = temp;
}
void fun2 (char** s1 , char** s2) {
   char* temp;
   temp = *s1;
  *s1 = *s2
  *s2 = temp ;
}
int main( ){
    char *str1 = “Hi”, *str2= “Bye”;
fun1(str1, str2);
printf(“%s %s”, str1, str2);
fun2(&str1, &str2);
printf(“%s, %s”, str1 , str2);
return 0;
}
```

The output of the program above is

(a) Hi Bye Bye Hi
(b) Hi Bye Hi Bye
(c) Bye Hi Hi Bye
(d) Bye Hi Bye Hi

28. [NAT] [GATE-2017 : 2M]

Consider the following C program.

```
#include<stdio.h>
#include<string.h>
void printlength(char *s, char *t) {
   unsigned int c=0;
   int len = ((strlen(s) – strlen(t)) > c) ? strlen(s) :
   strlen(t);
   printf("%d\n", len);
}
void main() {
   char *x = "abc";
   char *y = "defgh";
   printlength(x, y);
}
```

Recall that strlen is defined in string.h as returning a value of type size_t, which is an unsigned int. The output of the program is_____.

29. [MCQ] [GATE-2011 : 1M]

What does the following fragment of C program print?

```
char c[ ] = “GATE 2011”;
char *p = c;
printf (“%s”, p + p[3] – p[1]);
```

(a) GATE 2011 (b) E 2011

(c) 2011 (d) 011

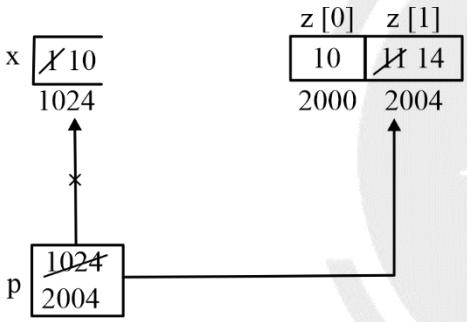
❑❑❑

ANSWER KEY

1.	(d)	**2.**	(a)	**3.**	(1023 to 1023)	**4.**	(c)
5.	(19 to 19)	**6.**	(81 to 81)	**7.**	(6 to 6)	**8.**	(10 to 10)
9.	(a)	**10.**	(d)	**11.**	(3 to 3)	**12.**	(2 to 2)
13.	(c)	**14.**	(a)	**15.**	(3 to 3)	**16.**	(30 to 30)
17.	(c)	**18.**	(2016 to 2016)	**19.**	(d)	**20.**	(140 to 140)
21.	(c)	**22.**	(d)	**23.**	(c)	**24.**	(d)
25.	(c)	**26.**	(a)	**27.**	(a)	**28.**	(3 to 3)
29.	(c)						

SOLUTIONS

1. (d)



→ *p=&x; stores the address of x into pointer p.

→ *p= 10; replaces value of x =1 by 10.

→ *p = &z[1]; stores the address of z[1] into pointer p.

Now,

&z [0] +1=&z [1]

(&z [0] +1) =*&z [1]

*(&z [0] + 1) = z [1]

*(&z [0] + 1) + 3 is same as z [1] + = 3

→ z [1] = z [1] + 3 = 11 + 3 =14

Because of statement *(&z[0] + 1) +=3, value at address 2004 is changed from 11 to 14.

Hence printf () prints 10, 10 and 14. Therefore correct option is (d).

Scan for Video solution

2. (a)

Given

$$\text{int a [3] [3] [3]} = \underbrace{\left\{\left\{\overbrace{\left\{1,2,3\right\}}^{a[0][0]}, \overbrace{\left\{4,5,6\right\}}^{a[0][1]}, \overbrace{\left\{7,8,9\right\}}^{a[0][2]}\right\}\right.}_{a[0]},$$

$$\underbrace{\left\{\left\{\overbrace{\left\{10,11,12,13,14,15,16,17,18\right\}}^{a[1][0]}\right\}\right.}_{a[1]},$$

$$\underbrace{\left\{\left\{\left\{19,20,21,22,23,24,25,26,27\right\}\right\}\right\}}_{a[2]}$$

i = 0, j = 0, k = 0,1,2

i	j	k	Printing
a[0]	[0]	[0]	1
a[0]	[0]	[1]	2
a[0]	[0]	[2]	3

i = 1, j = 0, k = 0,1,2

A → 65	Q → 81	a → 97
B → 66	R → 82	b → 98
C → 67	S → 83	c → 99
D → 68	T → 84	d → 100
E → 69	U → 85	.
F → 70	V → 86	.
G → 71	W → 87	.
H → 72	X → 88	.
I → 73	Y → 89	.
J → 74	Z → 90	.
M → 77		.
N → 78		x→ 120
O → 79		y→ 121
P → 80		z→ 122

i	j	k	Printing
a[0]	[0]	[0]	10
a[0]	[0]	[1]	11
a[0]	[0]	[2]	12

i = 2, j = 0, k = 0, 1, 2

i	j	k	Printing
a[0]	[0]	[0]	19
a[0]	[0]	[1]	20
a[0]	[0]	[2]	21

On verifying properly with options, option (a) is correct.

Scan for Video solution

3. (1023 to 1023)

Given that Z is an array which has index 0 to 9 and each index has value 1.

1	1	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9

Simple Function (Z, 10, 2) → calling function

↑

name of array

Array Z and array Y are same, because array is always call by reference.

As calling function is having Z, 10, 2 in parameters. n and x will take the values 10 and 2 respectively.

x = 2, n = 10

As initial value of Y[0] is 1, therefore initially value of variable total is 1.

Rewriting expression total = x × total + Y[i] for better understanding to

$total_i = x \times toal_{i-1} + Y[i]$, initially x = 2, each Y[i] = 1 and $total_0 = 1$

$\boxed{i=1}$ $total_1 = x \times total_0 + Y[1]$

$total_1 = 2 \times 1 = (2 + 1)$

$\boxed{i=2}$ $total_2 = x \times total_1 + Y[i]$

$total_2 = 2 \times (2 + 1) + 1 = 2^2 + 2^1 + 1$

$\boxed{i=3}$ $total_3 = 2 \times total_2 + Y[i]$

$= 2 \times (2^2 + 2^1 + 1) + 1$

$Total_3 = 2^3 + 2^2 + 2^1 + 1$

.

.

.

Similarly,

$\boxed{i=9}$ $total_9 = \underline{2^9 + 2^8 + 2^7 + \ldots..2^1 + 1}$

Above series is in GP with 10 terms

We know that

$$\boxed{\text{Sum of first n terms of a GP} = \frac{a(r^n - 1)}{r - 1}}$$

Where,

a = first term

r = common ratio

n = number of terms

Writing $2^9 + 2^8 + 2^7 + 2^6 + \ldots\ldots 2^1 + 1$ in reverse order we get,

$1 + 2^1 + 2^2 + 2^3 + \ldots. 2^9$

a = 1, n = 10, common ratio (r) = 2

Substituting in above formula

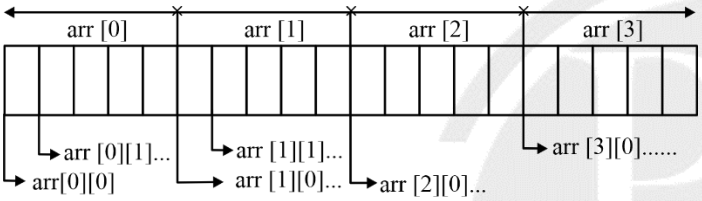
$$S_{10} = \frac{1(2^{10}-1)}{2-1}$$

= 1023 is the answer, which is the value of total$_9$.

Scan for Video solution

4. (c)

Given array int arr[4][5], it is read as arr is an array of 4 1-Dimensional array and each 1-Dimension array contains 5 elements

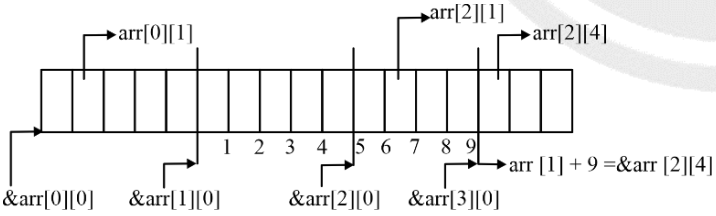


Note: array name represents the address of its first element.

Example: arr[1] = &arr[1][0]

arr[1] + 9 = &arr[1][0] + 9

(9 crepresenting to move 9 locations in forward direction from address of arr[1][0](&arr[1][0]))



arr[1] + 9 = &arr[2][4]

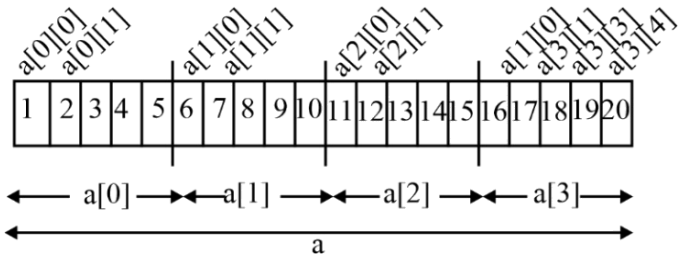
*(arr[1] + 9) = arr[2][4]

Therefore arr[2][4] = 10 * 2 + 4 = 24.

*(arr[1] + 9) = arr[2][4] = 24

Scan for Video solution

5. (19 to 19)



: array name (a is collection of 4 elements a[0],a[1], a[2],a[3])

Array name represent address of its first elemetns.

a = &a[0]

*a = *&a[0]

*a = a[0]\

What is a [0]?

a[0] is an array of 5 elements i.e a[0][0], a[0][1], a[0][2] & a[0][3],a[0][4]

a[0] ⇒ address of its first element

a[0] = &a[0][0]

*a = a[0]

*a = a[0] = &a[0][0]

*a = &a[0][0]

**a = a[0][0]

$\boxed{a[0][0] = 1}$

*(a + **a + 2) = *(a + 1 + 2) = * (a + 3) = a[3]

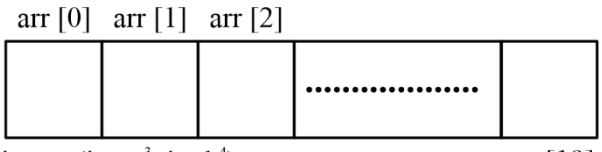
*(a + **a + 2) = a[3]

*(a+ **a+2) + 3 = a[3]+3

*(*a+ **a + 2) + 3) = a[3][3] = 19

Scan for Video solution

6. (81 to 81)



int pp (int. a[3], int b[4])

tot = 1 ex = 3

Note:- that arr[i] in tob is same as arr[i] in pp and also array contains garbage value initially.

Understanding tob function:

b = 4 2

0	0	1	G	G	G	G
arr [0]	arr [1]	arr [2]	arr [3]	.....		arr [19]

if (b%2) arr[i] = 1; exeutes when b is odd

else arr[i] = 0; exeutes when b is even

b = b/2 exeutes every time

(1) i = 0, b > 0 ⇒ 4 > 0 ⇒ true(b is even)
arr[i] = 0
arr[0] = 0
b = b/2 = 4/2 = 2

(2) i =1 b > 0 ⇒ 2 > 0 ⇒ true(b is even)
arr[i] = 0
arr[1] = 0
b = b/2 = 2/2 = 1

(3) i = 2, b > 0 ⇒ 1 > 0 ⇒ true (b is odd)
arr[i] = 1
arr[2] = 1
b = b/2 = ½ = 0

(4) i = 3, b > 0 ⇒ 0 > 0 ⇒ false

Here loop terminates.

Understanding pp function:

ex: = 3 9 81

tot = 1 81

```
len = 3
for(i = 0;i < len;i++)
{
if(arr[i]==1)
tot=tot*ex;
ex=ex*ex;
}
```

above code is executed from i=0 to 2(len =3)

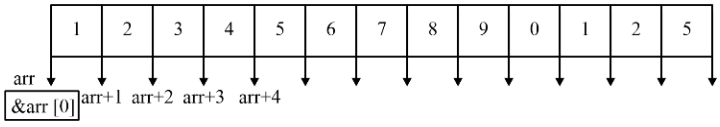
(i) i=0
arr[i]= =1
0= =1(false)
ex=ex*ex
ex=3*3
ex=9

(ii) i=1
arr[i]==1
0==1(false)
ex=ex*ex
ex=9*9=>81
ex=81

(iii) i=2
arr[i]==1
1==1(true)
tot=tot*ex
tot=1*81
tot=81
ex=ex*ex
ex=81*81
81 is returned.

Scan for Video solution

7. (6 to 6)



arr is name of the given array, we know that array name represents the address of first element.

arr=&arr[0]

arr+4=&arr[0]+4(moving 4 locations ahead from &arr[0])

arr+4 = &arr[4]

ip=&arr[4]

ip[1] = *(ip+1) = *(&arr[4]+1)

ip[1] = *(&arr[5])

ip[1] = arr[5]

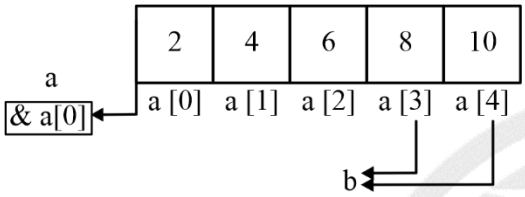
ip[1] = 6

Hence 6 is the correct answer

Scan for Video solution

8. (10 to 10)

Constructing the array according to the given data.



Initially, value of sum = 0

b = a + 4

b = &a[0] + 4

b = &a[4]

*b = a[4]

b-0 = &a[4]

b-1 = &a[3]

b-2 = &a[2]

b-3 = &a[1]

b-4 = &a[0]

Note: Scope of for loop remains till the first semicolon (;).

i = 0 sum = sum + (*b – 0) – *(b – 0)

sum=0 + (10 – 0) – 10 = 0

i = 1 sum=0 + (10 – 1) – 8 = 1

i = 2 sum=1 + (10 – 2) – 6 = 3

i = 3 sum=3 + (10 – 3) – 4 = 6

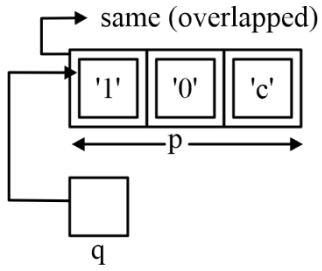
i = 4 sum=6 + (10 – 4) – 2 = 10

Finally, value which is stored in sum variable is printed,

Therefore 10 is printed.

Scan for Video solution

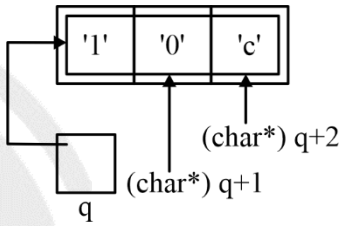
9. (a)



'a' + 2⇒ 'c'

Intitally q is pointing to address of whole structure variable p.

(char*)q : Here typecasting is being done,q is holding address of a character i.e q is pointing to addresses of '1' (First member)



(char*) q : Address of char '1'

(char*) q + 1 : Address of next character i.e. '0'

(char) q+1: '0'

(char*) q+2 : Address of 'c'

(char)q+2: 'c'

printf("%c",*((char*)q+1), *((char*)q+2)); will print 0,c without any quotes. Hence option A is correct.

Scan for Video solution

10. (d)

A (false): Compiler error as the return of malloc is not type casted appropriately.

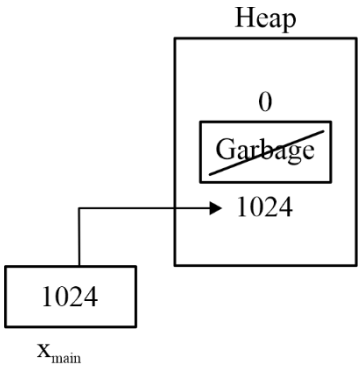
False because no typecasting is required. In C++, it is required but not in C language.

B(false): Compiler error because the comparison should be made as x==NULL and not as shown.

False, as we know that NULL==x or x==NULL are same, hence option B is also false.

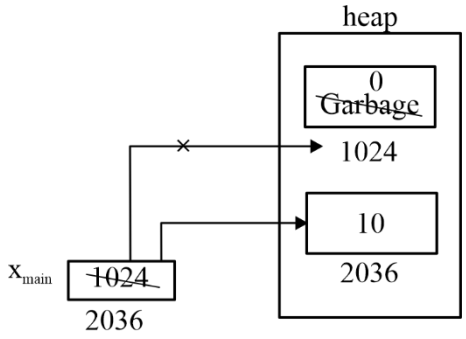
C(false): Compiles successfully but execution may result in dangling pointer.

```
void main(){
int *x = malloc(sizeof(int));
if(NULL==x)
return; //if no memory is available then return.
```




```
assignval (1024, 0) is called
xlocal = 1024
val= 0
xlocal =0
value at memory location pointed by xlocal = 0
value at memory location 1024 = 0
return x, returns 1024 to x in main.
if(x){
x=(int*)malloc(sizeof(int))
if(NULL==x)
return;
x=assignval(x,10);
}
printf("%d",*x);
free(x);
}
```

Note that code written in solution are just used to represent.



we are again allocating memory without freeing old memory, x is now pointing to memory location 2036 and we cannot access to memory location 1024 which is lost (also known as memory leak problem), and also there is no pointing of memory location which is deleted, hence there is no dangling pointer problem. Therefore, above code suffers from memory leak problem, hence option D is correct answer.

Scan for Video solution

11. (3 to 3)

Firstly, understanding the first loop in the given program.

Initial value of done variable and content of the array shown below.

```
while (done = = 0) {
done =1;
for(i=0;i<=4;i++){
if(array[i]<array[i+1]{
swap(&array[i], &array[i+1]);
done= 0;
```

done

~~0~~ 1 0

3	5	1	4	6	2

(i) done = = 0 (true)

code inside while loop will execute

(i) done = = 1
(ii) for loop will execute from i = 0, 1, 2, 3, 4

Executing for loop

(1) i = 0

we are comparing element of i^{th} index and $i+1^{th}$ index

array[0]<array[1] →3<5 true, swapping will take place.

swap(&array[0],&array[1])

5	3	1	4	6	2
0	1	2	3	4	5

Again, done variable value becomes 0

(2) i = 1

array [1]< array [2] →3<1 false, no swapping will take place.

(3) i = 2

array [2]< array [3] →1<4 true, swapping will take place.

5	3	4	1	6	2
0	1	2	3	4	5

done = 0 (No effect because done is already 0)

(4) i = 3

array [3]< array [4]→1<6 true, swapping will take place.

5	3	4	6	1	2
0	1	2	3	4	5

done =0(no effect)

(5) i = 4

array[4]<array[5] true, swapping will take place

5	3	4	6	2	1
0	1	2	3	4	5

⇒ 1st loop terminate

Now understanding 2nd loop of the given program.

done = 0

5	3	4	6	2	1
0	1	2	3	4	5

```
for(i= 5;i> =1;i--){
if(array[i] >array [i– 1]){
swap(&array [i], &array [i –1]);
done =0;
  }
}
```

i = 5 array [5] > array [4] ⇒ 1 > 2 false, now swap

i = 4 array [4] > array [3] ⇒ 2 > 6 false, no swap

i = 3 array [3] > array [2] ⇒ 6 > 4 (true), swap

5	3	6	4	2	1
0	1	2	3	4	5

done = 0 (No effect)

i = 2 array [2] > array [1] ⇒ 6 > 3 (true), swap

5	6	3	4	2	1
0	1	2	3	4	5

done = 0 (No effect)

array [1] > array [0] ⇒6 > 5→ true, swap will take place.

6	5	3	4	2	1
0	1	2	3	4	5

done = 0 (No effect)

(5) i = 1

Both the loop ends

done

0	6	5	3	4	2	1
	0	1	2	3	4	5

```
While(done==0)
{
 done = 1;
      1st loop
      2nd loop
      }
```

Another iteration of while loop takes place

Value of done becomes 1.

1st loop will execute for i = 0, 1, 2, 3, 4

6	5	3	4	2	1
0	1	2	3	4	5

i = 0 array [0] < array [1] ⇒ false, no swap

i = 1 array [1] < array [2] ⇒ false, no swap.

i = 2 array [2] < array [3] ⇒ true, swap

6	5	4	3		2	1
0	1	2	3		4	5

done = 0 is made

Array is sorted in decreasing order now i.e. arra y [1] > array [i +1], As this is true that means array [i] < array [i +1] is false for all i, 1st loop is terminated.

6	5	4	3	2	1

done = 0

2nd loop : because array is sorted in decreasing order.

$\Rightarrow$ array [i] < array [i –1]

i.e. array [i] > array [i –1] is false for all i.

Hence 2nd loop terminates

while (done = = 0)

0 = = 0 (true)

Another iteration of while loop is executed.

done

Ø 1	6	5	4	3	2	1
	0	1	2	3	4	5

1st loop, no change because array is already sorted.

2nd loop, no change in array because of same above reason.

While (done = = 0)

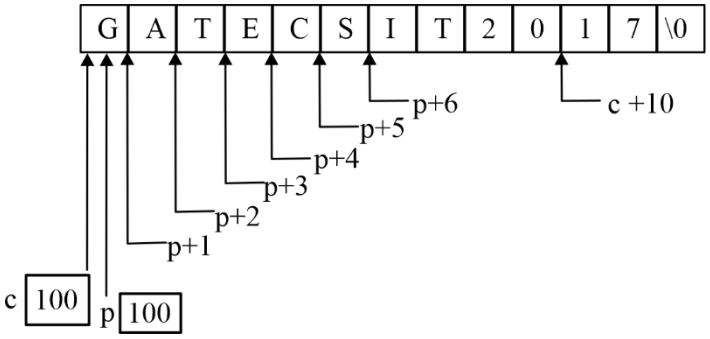
1 = = 0(False) while loop terminates.

Finally, array [3] is printed, therefore 3 is printed as output

Scan for Video solution

12. (2 to 2)

```
int  main ()
{
char *c = "GATECSIT2017";
char *p = c;
printf ('%d', (int) strlen(c +2 [p] – 6 [p] –1));
return 0;
}
```



p + 2 = moving 2 locations in forward direction from address of 'G'.

c is holding base address (address of G)

p + 2 = Address of 'T'

$\boxed{*(p+2)='T'}$

$\boxed{p[2]=2[p]=*(p+2)=*(2+p)='T'}$

p + 6 = Address of 'I'

* (p + 6) = 'I'

$\boxed{p[6]=6[p]=*(p+6)=*(6+p)='I'}$

Example:

c + 'B' – 'A': What are value for 'B', 'A' Here ASCII values are used.

c + 66 – 65

= c + 1; address of 'A'

c + 2 [p] – 6 [p] –1
= c + 'T' – 'I' –1
= c + ~~x~~ + 11 – ~~x~~ – 1
= c + 10

I	$\rightarrow$ x
J	$\rightarrow$ x + 1
K	.
L	.
M	.
N	.
O	.
P	.
Q	.
R	.
S	.
T	$\rightarrow$ x + 11

c + 10 $\Rightarrow$ Address of 'G' + 10
= Moving 10 location in forwards direction from address of 'G'

c + 10 $\Rightarrow$ address of char '1'.

c + 10 $\Rightarrow$ represents string "17"

strlen (c + 10) = 2

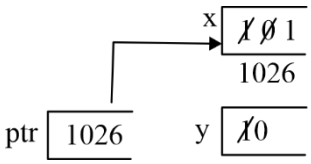
Note: strlen returns unsigned integer

Therefore 2 is returned.

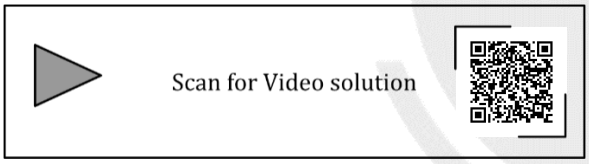
Scan for Video solution

13. (c)

```
void printxy (int x int y)
{
int *ptr;
x =0;→ will make x as 0
ptr =&x; → assign 1026 to ptr
y = *ptr;→ y = value at (memory location 1026)
            y = 0
*prt = 1;→ value at memory location 1026 = 1)
printf ("%d%d" x,y);
}
printxy (1   1)
          ↑ ↑
          x  y
```



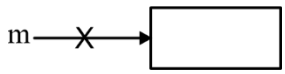
lastly value of x and y is printed as 1, 0; Hence option (c) is correct answer.



14. (a)

(P) static char var: Stored in data segment
P – ii

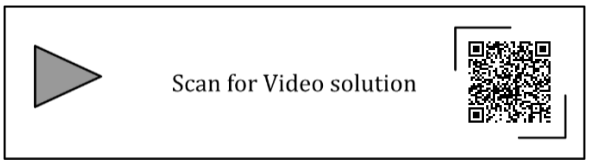
(Q) m = malloc (10);
m = NULL;



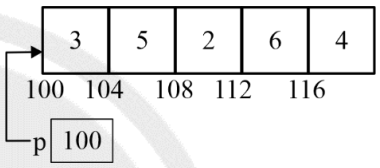
No way to access memory location i.e.
Lost memory
Q → IV

(R) char *ptr [10]
ptr is an array of 10 pointers to chaaracter i.e. 10 pointer can hold 10 address.
R → (i)

(S) register int var 1
Register variable is only a requests /recommendation
S → (iii)
P→ (ii), Q→ (iv), R → (i), S →(iii)
Hence, option A is correct.



15. (3 to 3)



p + 1 = moving 1 location in forward direction from address holding pointer variable p.

p + 1 = memory location 104

⇒*(p + 1) = Then it would be value at memory location 104

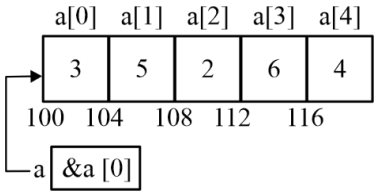
*(p + 1) = 5

p[1] = 5

NOTE: If p is pointing to address of integer, then

p[0]: is value at the address

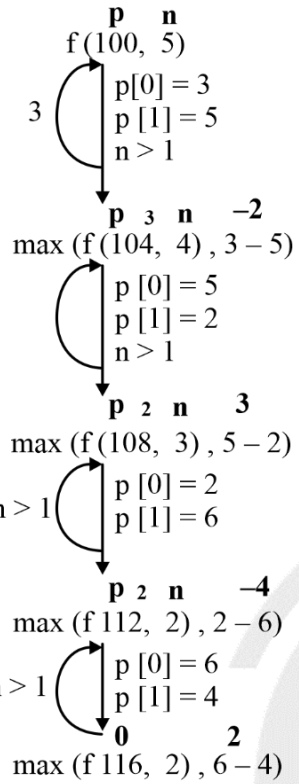
p[1]: is value at next address



NOTE: array name always points to the address of first index of an array.

f(a,5) → f(100,5)

max(f(p+1),n-1,p[0]-p[1]) will be executed whenever n>1



f(116,1) ⇒ n<=1 is true, returns 0.

Hence 3 is returned.

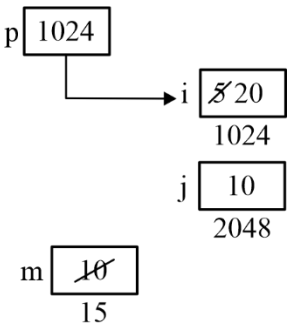
Scan for Video solution

16. (30 to 30)

f(&i, j) → f (1024, 10)

↑ ↑

(Address, value)



*p = *p + m

(value at memory 1024 + 15)

*p = 5 + 15 = 20

Hence, 20 + 10 = 30 is printed

Scan for Video solution

17. (c)

a = 3 (declaring new variable)

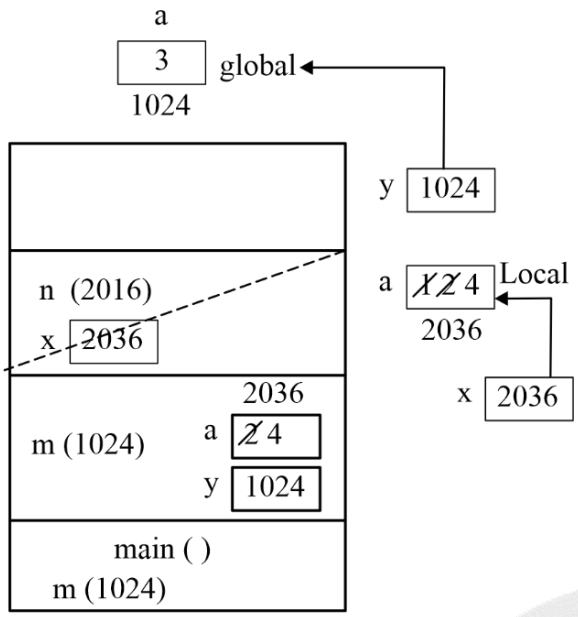
a = 1(declaring new variable)

a=3(Global scope, because it is not inside any function)

a=1(local scope, because declared inside m())

As it is mentioned in question that it is using dynamic scoping and pass by reference, rewriting the code for better understanding.

```
int a=3;
void n(int(*x)){
*x=*x*a;
print(*x);
}
void m(int *y){
int a=1;
a=*y-a;
n(&a);
print(a);
}
void main(){
m(&a);//there is no local a so, a is global variable
```



a = y –a (update local a)
a = value at (1024) –1
a = 3 – 1 = 2

a = y –a (update local a)

a = value at (1024) –1

a = 3 – 1 = 2

at n(&a), n(2036) is called

*x=*x*a;

= value at(2036)*a(here, previous call will be resolved)

i.e…..a of m()

=2 * 2 = 4 i.e….value at (2036) = 4

Lastly 4 is printed (value of x)

As function n gets deleted from activation record, updating has no meaning, m() will also print 4 and gets deleted from activation record and lastly main also gets deleted from activation record.

Therefore, C is correct option.

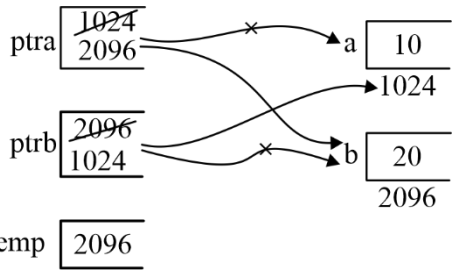
Scan for Video solution

18. (2016 to 2016)

Let mystery (&a, &b) is called

mystery $\left(\overset{ptra}{1024}, \overset{ptrb}{2096}\right)$

Note:-No dereferencing is taking palce

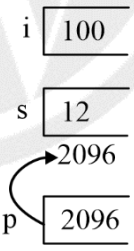


As soon as execution of this function ends, there is nothing available in activation record, no change in actual argument ie…. mystery function is not making any changes in actual argument. Hence actual value of a is printed ie…2016

Scan for Video solution

19. (d)

Function is expecting 1st argument as int type and 2nd argument as short type.



Option (A):

f(s, *s)

1st agreement is short type but function is expecting this argument to be type of int, but because of implicit conversion, short is converted into int. so 1st argument will not cause any error 2nd argument is pointer but function is expecting it to be short type which will generate an error.

∴ (A) is wrong

Option B

i = f(i,s)

Return type is not void, hence option (b) is also wrong.

Option C:

f(i, *S)

1st argument is int but 2nd argument is pointer i.e. There exists system error. So option c is also wrong. (s is not holding any address.)

Option D:

f(i, *p)

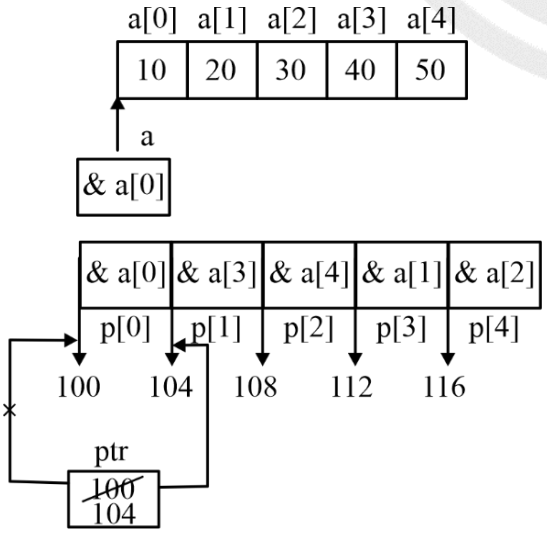
Both types are matching

First argument is int and p is pointer to short type, Hence, (f) is correct.

s is not (a) pointer.

20. (140 to 140)

static int * p[] is read as p is an array of pointer to integer each element of p[ ] holds an address of integer type variable.



(array name) a = &a[0]

a +1 = moving one location in forward direction from &a[0] = &a [1]

NOTE: Array name represents address of its first element.

a + 2 = & a [2]

a + 3 = & a [3]

a + 4 = & a [4]

int ** ptr = p;

Here p is array name, which represents address of its first elements.

ptr++=ptr will point to next address or location ie….104

(i) &p[1] – &p[0]

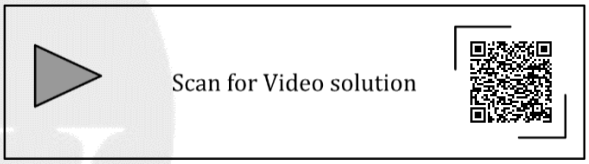
$$= \frac{\text{Actual difference}}{4} = \frac{104 - 100}{4} = 1$$

(ii) ** ptr

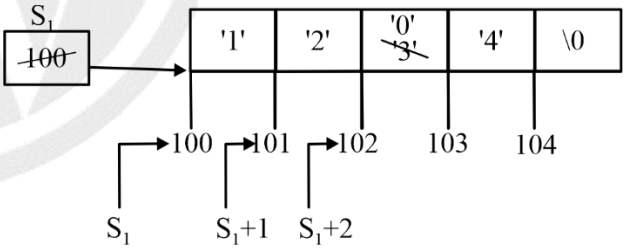
**(& p [1])

= ** &p[1] =*p[1] =*&a[3] = a[3] = 40

Hence, 140 is printed.



21. (c)



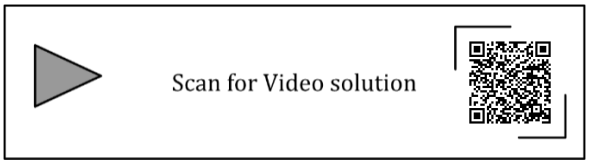
p = s1 + 2

⇒ Address + 2 (moving 2 locations in forward direction)

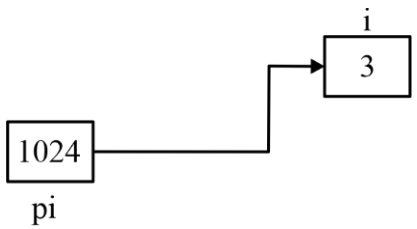
p = 102 (address of '3')

* p = '0' (Assign character '0' (zero) at address 102)

Hence, 1204 is printed, option c is correct.



22. (d)



scanf("%d", pi)

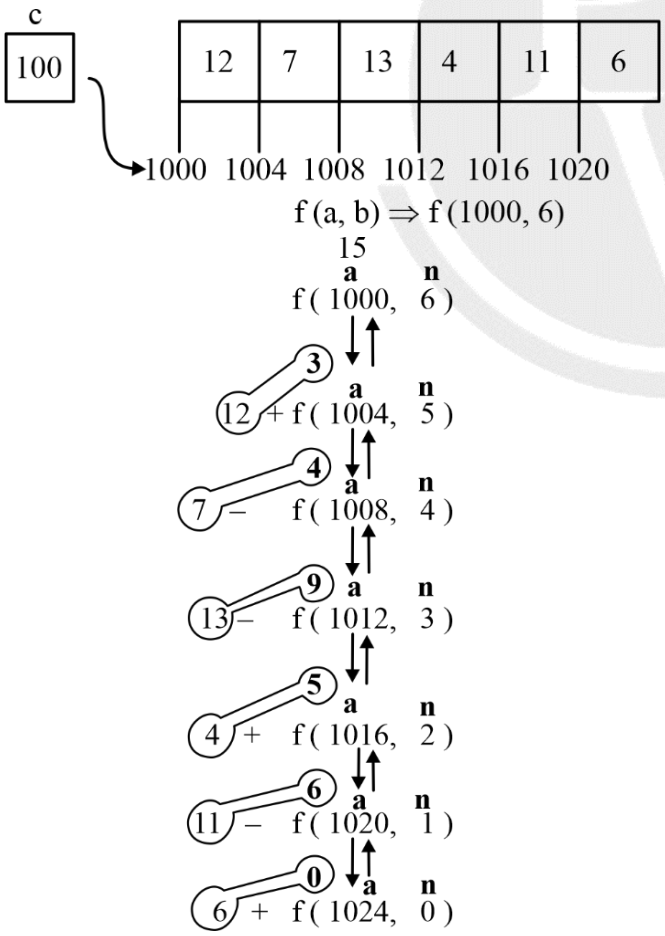
scanf takes address as input, whatever the input given will be stored at memory location 1024.

Let the input is 3

The printed value is 5 more than the input entered therefore option (d) is correct answer.

Scan for Video solution

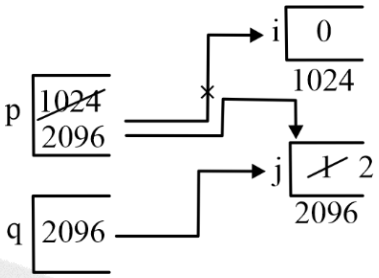
23. (c)



Hence 15 is printed and option (c) is correct answer

Scan for Video solution

24. (d)



Both p and q are pointing to j (2096)

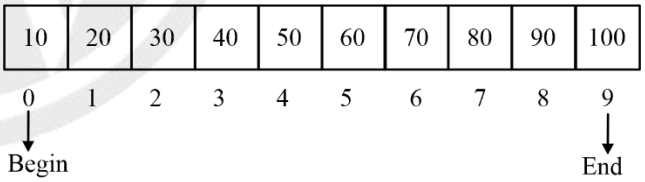
* p = 2 that means insert value at address 2096

0, 2 is printed hence, correct option is(d).

Scan for Video solution

25. (c)

Firstly, let us understand how binary search works.



Case: 1

Let us suppose we are searching for x =90

$$\text{Mid} = \frac{\text{Begin} + \text{end}}{2} = \frac{0+9}{2} = 4$$

$A[\text{mid}] = = x$

↓ No ↓

40 90

x is greater than elements at Mid position.

(1) x cannot be present from any location between Begin and mid new list must be from Mid +1 to end index.

Begin = Mid + 1

Begin = 5

Again, Mid = $\frac{(5+9)}{2} = 7$

Check A [Mid] = = x

A [7] = = 90 (No)

A [Mid] < x

Update ⇒ Begin = Mid + 1 = 8

Mid = $\frac{8+9}{2} = 8$

Check A [Mid] = x

yes

break and return.

Case: 2

Begin = 0, End = 9, x = 120

(i) Mid = $\frac{0+9}{2} = 4$

A [Mid] = = x No

(ii) Begin = Mid + 1 = 5

Mid = $\frac{5+9}{2} = 7$

A [7] = = x No

A [Mid] < x

(iii) Begin = Mid + 1 = 8

Mid = $\frac{8+9}{2} = 8$

A [Mid] = = x No

A [Mid] < x

(iv) Begin = Mid + 1 = 9

Mid = $\frac{9+9}{2} = 9$

A [Mid] = = x No

A [Mid] < x

Begin = Mid + 1 = 10

Begin becomes greater than End, unsuccessful search, element is not present.

Repeat till Begin < End && A [Mid]! = x

Case: 3

x = = 40

(i) Begin = 0, End = 9

Mid = $\frac{0+9}{2} = 4$

A [Mid] = = x No

x < A [Mid]

⇒ x cannot be present from Mid to End position.

New list must be from Begin to Mid –1

End = Mid –1

(i) We need to calculate mid and check every time if element is equal then search result is found.

(ii) If element is small then Mid search should be performed on left side.

(iii) If x is bigger than Mid then search should be performed on right side of list.

Now, focusing on the actual code.

(i) k = $\frac{i+j}{2}$ is performing mid operation

(ii) If (Y[k] < x) i = k; else j = k;

- In above code, if element to be searched is larger than Mid, then there should be i = k +1 (Begin = Mid +1) but above it is being assigned without incrementing.
- And if element is smaller then j = k – 1 (End = Mid–1) must have been performed but above it is being assigned without decrementing in else part.

The above given code will not work if

(i) x (Element to be searched) is greater than the maximum element (last element)

(ii) Or we are searching for the last element itself.

Case: 4

10	20	30	40	50	60	70	80	90	100
0	1	2	3	4	5	6	7	8	9
↓ i								↓ j	

(i) Let us assume we are searching x = 100

j = 9

k = $\frac{0+9}{2} = 4$

Y [4] < x (True)

i = k ⇒ i = 4 (i and k are printing same index)

(ii) $k = \frac{4+9}{2} = 6$

Y [6] < x (True)

i = k ⇒ i = 6

(iii) $k = \frac{6+9}{2} = 7$

Y [7] < x (True)

i = k ⇒ i = 7

(iv) $k = \frac{7+9}{2} = 8$

Y [8] < x (True)

i = k ⇒ i = 8

(v) $k = \frac{8+9}{2} = 8$

y [8] < x (True)

i = k ⇒ i = 8

i remains same (no update)

Program goes in to infinite loop, i will never become equal to j or greater then j.

∴ As above example is illustrated in (case 4), therefore correct option is option c, because program goes int. infinite loop.

Scan for Video solution

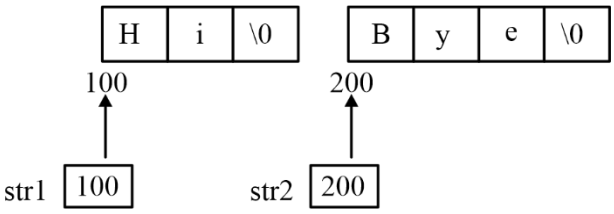
26. (a)

- i = k has to be replaced by i = k + 1 if elements is larger than Mid (k is Mid).
- j =k has to be replaced by j = k – 1 if elements is smaller than Mid.

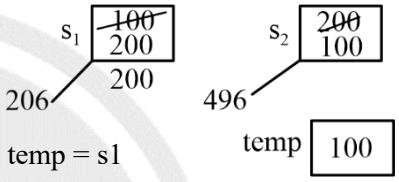
Hence, option A is correct answer.

Scan for Video solution

27. (a)



fun1 () is called, fun1 (100, 200)

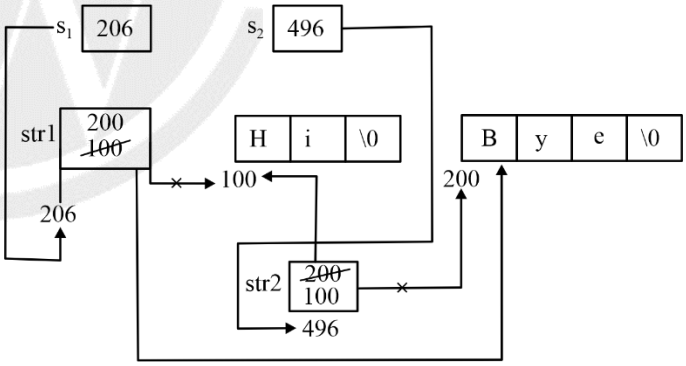


$s_1 = s_2$

s_2 = temp

No change in str1 and str2, so HiBye.. is printed from the first printf statement.

Now fun2 () is called, fun2(206, 496)



temp = *s1 (value of at memory location 206)

temp = 100

* $s_1 = *s_2$ (value at memory location 496)
* s_1 = 200
* s_2 = temp
* s_2 = 100

- contents of str1 and str2 are swapped
- Now str1 is pointing to 'Bye' and str2 is pointing to "Hi"

Therefore 2nd printf prints ByeHi.

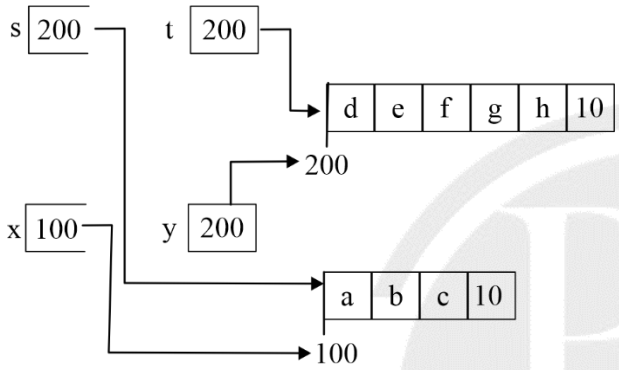
Hence (a) is correct answer.

Scan for Video solution

28. (3 to 3)

char*x = "abc";→ Read only area

printlength (x, y); → printlength(100,200)



strlen(s) – strlen(t)

3 – 5 = –2

Unsigned int – unsigned int

Will also be an unsigned int

⇒ – 2 is treated as max integer value – 2

= large +ve value

$len = (very\ large\ positive\ val > 0)\ ?\ strlen(s)^{3} : strlen(t)^{5}$

true

Hence 3 is printed.

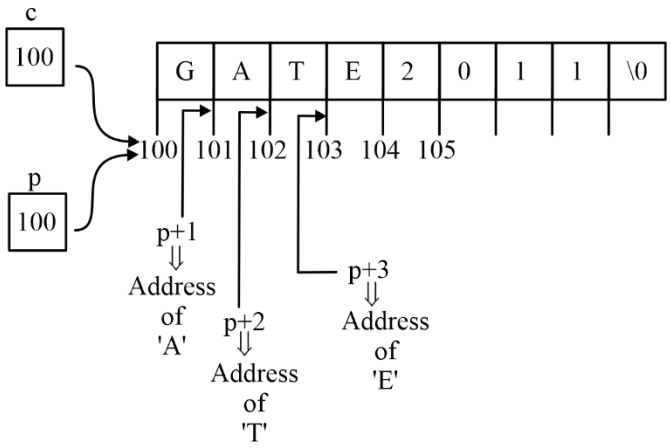
Scan for Video solution

29. (c)

char c [ ] = "GATE 2011";

char *p= c;

printf ("%s", p + p [3] – p[1])



p+ 1 ⇒ Address of 'A'

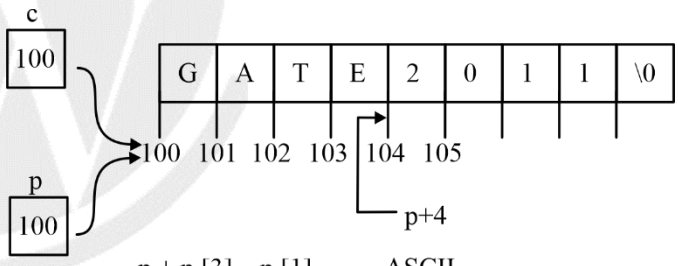
* (p +1) ⇒ 'A'

$\boxed{p[1] = 'A'}$

p + 3 ⇒ Address of 'E'

* (p +3) ⇒ 'E'

$\boxed{p[3] = 'E'}$



p + p [3] – p [1]
= p + 'E' – 'A'
= p + x + 4 – x
= p + 4
= Address 104

ASCII
A → x
B → x +1
C → x +2
D → x +3
E → x +4

2011 is printed till \0, therefore option c is correct.

Scan for Video solution

❑❑❑

CHAPTER

5 ARRAYS AND LINKED LIST

Arrays

1. [NAT] [GATE-2015: 2M]

Suppose c = ⟨c[0],… , c[k-1] ⟩ is an array of length k, where all the entries are from the set {0,1}. For any positive integers a and n, consider the following pseudocode.

DOSOMETHING (c, a, n)

Z ← 1

for i ← 0 to k-1

do z ← z^2 mod n

if c[i] = 1

then z ← (z × a) mod n

return z

If k = 4, c = ⟨1,0,1,1⟩, a = 2 and n = 8, then the output of DOSOMETHING (c, a, n) is _____

2. [MCQ] [GATE-2014: 1M]

Let A be a square matrix of size n × n. Consider the following pseudo code. What is the expected output?

```
C = 100;
for i = 1 to n do
      for j = 1 to n do
   {
      Temp = A[i][j] + C;
      A[i][j] = A[j][i];
      A[j][i] = Temp – C;
      }
for  i = 1 to n do
      for j = 1 to n do
      Output  (A[i][j]);
```

(a) The matrix A itself

(b) Transpose of the matrix A

(c) Adding 100 to the upper diagonal elements and subtracting 100 from lower diagonal elements of A

(d) None of the above

3. [MCQ] [GATE-2014: 2M]

Consider the following C function in which size is the number of elements in the array E:

```
int MyX(int * E, unsigned int size) {
      int Y = 0;
      int Z;
      int i, j, k;
     for (i = 0; i < size; i++)
    Y = Y + E[i];
for (i= 0; i < size; i++)
  for (j = i; j< size; j++)   {
     Z = 0;
     for (k = i; k <= j; k++)
                          Z = Z + E[k];
     if (Z>Y)
                          Y = Z;
        }
return Y;
}
```

The value returned by the function MyX is the-

(a) maximum possible sum of elements in any sub-array of array E.

(b) maximum element in any sub-array of array E.

(c) sum of the maximum elements in all possible sub-arrays of array E

(d) the sum of all the elements in the array E.

Sorting Arrays

4. [MCQ] [GATE-2015: 2M]

Consider the following two C code segments. Y and X are one and two dimensional arrays of size n and $n \times n$ respectively, where $2 \le n \le 10$. Assume that in both code segments, elements of Y are initialized to 0 and each element X[i][j] of array X is initialized to i + j. Further assume that when stored in main memory all elements of X are in same main memory page frame.

Code segment 1:

```
// initialize elements of Y to 0
// initialize elements X [i][j] of X to i + j
    for (i = 0; i < n; i++)
    Y[i] += X[0][i];
```

Code segments 2:

```
// initialize elements of Y to 0
// initialize elements X[i][j] of X to i + j
    for (i = 0; i < n; i + +)
     Y[i] += X[i][0];
```

Which of the following statements is/are correct?

S1: Final contents of array Y will be same in both code segments.

S2: Elements of array X accessed inside the for loop shown in code segment 1 are contiguous in main memory.

S3: Elements of array X accessed inside the for loop shown in code segment 2 are contiguous in main memory.

(a) Only S2 is correct

(b) Only S3 is correct

(c) Only S1 and S2 are correct

(d) Only S1 and S3 are correct

Linked List

5. [MCQ] [GATE-2023 : 1M]

Let SLLdel be a function that deletes a node in a singly-linked list given a pointer to the node and a pointer to the head of the list. Similarly, let DLLdel be another function that deletes a node in a doubly-linked list given a pointer to the node and a pointer to the head of the list.

Let n denote the number of nodes in each of the linked lists. Which one of the following choices is TRUE about the worst-case time complexity of SLLdel and DLLdel?

(a) SLLdel is O(1) and DLLdel is O(n)

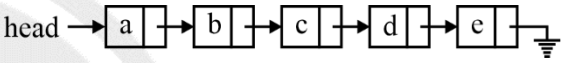
(b) Both SLLdel and DLLdel are O(log(n))

(c) Both SLLdel and DLLdel are O(1)

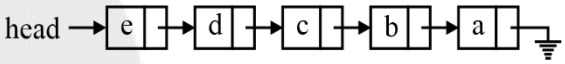
(d) SLLdel is O(n) and DLLdel is O(1)

6. [MCQ] [GATE-2022: 1M]

Consider the problem of reversing a singly linked list. To take an example, given the linked list below,



the reversed linked list should look like



Which one of the following statements is TRUE about the time complexity of algorithms that solve the above problem in O(1) space?

(a) The best algorithm for the problem takes $\theta(n)$ time in the worst case.

(b) The best algorithm for the problem takes $\theta(n \log n)$ time in the worst case.

(c) The best algorithm for the problem takes $\theta(n^2)$ time in the worst case.

(d) It is not possible to reverse a singly linked list in O(1) space.

7. [MCQ] [GATE-2021: 2M]

Consider the following ANSI C program:

```
#include <stdio.h>
#include <stdlib.h>
struct Node{
int value;
struct Node *next;};
int main( ){
```

```
struct Node *boxE, *head, *boxN; int index = 0;
boxE = head = (struct Node *) malloc(size of(struct Node));
head → value = index;
for (index = 1; index <= 3; index++){
boxN = (struct Node *) malloc(size of(struct Node));
boxE → next = boxN;
boxN → value = index;
boxE = boxN; }
for (index = 0; index ≤ 3; index++) {
printf ("Value at index %d is %d\n", index, head → value);
head = head → next;
printf ("Value at index %d is %d\n", index+1, head → value); } }
```

Which one of the statements below is correct about the program?

(a) Upon execution, the program creates a linked-list of five nodes.

(b) Upon execution, the program goes into an infinite loop.

(c) It has a missing return which will be reported as an error by the compiler.

(d) It dereferences an uninitialized pointer that may result in a run-time error.

8. [MCQ] [GATE-2017: 1M]

Consider the C code fragment given below.

```
typedef struct node
{ int data;
node* next;
} node;
void join (node *m, node *n){
node *p = n;
while (p → next ! = NULL){
p = p → next;
}
p → next = m;
}
```

Assuming that m and n point to valid NULL-terminated linked lists, invocation of join will

(a) append list m to the end of list n for all inputs.

(b) either cause a null pointer dereference or append list m to the end of list n.

(c) cause a null pointer dereference for all inputs

(d) append list n to the end of list m for all inputs

9. [MCQ] [GATE-2014: 2M]

What is the worst case time complexity of inserting n elements into an empty linked list, if the linked list needs to be maintained in sorted order?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^2)$ (d) $\Theta(1)$

10. [MCQ] [GATE-2010: 1M]

The following C function takes a singly-linked list as input argument. It modified the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

```
typedef struct node{
int value;
struct node * next;
}Node;
Node * move_ to _ front (Node * head){
Node * p, *q;
if ((head = = NULL)|| (head→next = =NULL))
return head;
q = NULL;
p = head;
while (p →next ! = NULL){
q = p;
p = p → next;
}
```

_____

```
return head;
}
```

Choose the correct alternative to replace the blank line.

(a) q = NULL ; p → next = head ; head = p;

(b) q→ next = NULL; head = p; p → next = head;

(c) head = p; p → next = q; q→ next = NULL;

(d) q→ next = NULL; p→ next = head; head = p;

11. [MCQ] [GATE-2008: 2M]

The following C function takes a singly-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1,2,3,4,5,6,7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node{
int value;
struct node * next;
};
void rearrange (struct node * list){
struct node * p, *q;
int temp;
if(!list||!list→next) return;
p=list; q=list->next;
while (q){
temp = p→value;
p→value = q→value;
q→ value = temp;
p = q→next;
q = p? p→next: 0:
}
}
```

(a) 1, 2, 3, 4, 5, 6, 7 (b) 2, 1, 4, 3, 6, 5, 7

(c) 1, 3, 2, 5, 4, 7, 6 (d) 2, 3, 4, 5, 6, 7, 1

❑❑❑

ANSWER KEY

1.	(0 to 0)	**2.**	(a)	**3.**	(a)	**4.**	(c)
5.	(d)	**6.**	(a)	**7.**	(d)	**8.**	(b)
9.	(c)	**10.**	(d)	**11.**	(b)		

SOLUTIONS

1. (0 to 0)

$Z \leftarrow 1$

for $i \leftarrow 0$ to $k-1$

do $z \leftarrow z^2 \bmod n$

If (c [i] = 1)

Then $z \leftarrow z \times a \bmod n$

return z

k = 4 c = [0,0,1,1], n = 8, a = 2;

(i) $i = 0, z = z^2 \bmod n \Rightarrow z = 1^2 \bmod 8 = 1$

if (c [i] =1) $\Rightarrow$ true because c [0] =1

$z = z \times a \bmod n$

$z = 1 \times 2 \bmod 8$

$z = 2$

(ii) $i = 1, z = z^2 \bmod n = 4$

if (c [i] = 1) $\Rightarrow$ false because c[1] =0

(iii) $i = z, z = 4^2 \bmod 8 = 0$

if (c [i] =1) $\Rightarrow$ true

No need to check anything further because z is 0 and all the operations involves multiplication of z.

$z = z^2 \bmod n$ & $z = 2 \times a \bmod n$. Because, z will never change from 0.

Scan for Video solution

2. (a)

$$A = \begin{array}{c} \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} 3 & 4 \\ 7 & 9 \end{bmatrix} \end{array} n = 2$$

```
for (i = 1; i < 2; i ++)
{
for (j = 1; j ≤ 2;  j++)
{
    temp = A [i] [j] + C;
    A [i] [j] = A [j] [i];
    A [j][i] = temp – C;
}
}
```

(i) if i = 1 for j = 1

temp = A [1] [1] + 100 = 103

A [1] [1] = A [1] [1]

A [1] [1] = 103 – 100 = 3

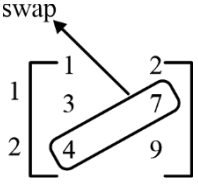
No effect on A_{11}

for j = 2

temp = A [1] [2] + 100 = 104

A [1] [2] = 7

A [2] [1] = 104 – 100 = 4



New matrix: A_{12} swapped with A_{21}.

(ii) i = 2

for j = 1

Temp = A [2] [1] + 100 = 104

A [2] [1] = A [1] [2] = 7

A [1] [2] = 104 – 100 = 4

Temp = 9 + 100 = 109

Matrix = $\begin{bmatrix} 3 & 4 \\ 7 & 9 \end{bmatrix}$

Again, A_{21} swapped with A_{12}.

for j = 2

temp = 9 + 100 = 109

A [2] [2] = A [2] [2]

A [2] [2] = 109 – 100 = 9

No effect.

Finally, we get the same matrix

- A_{ij} is swapped with its mirror image A_{ji} (Lower triangular gular/upper triangular)
- And when the code executes for the mirror image again both elements are swapped

⇒ Same matrix

3. (a)

```
for (i = 0; i < size; i++)
// find the sum of all elements of the array E and
storing the results in Y.
for (i = 0; i < size; i++)
for (j = i; j < size; j++)
```

- i = 0

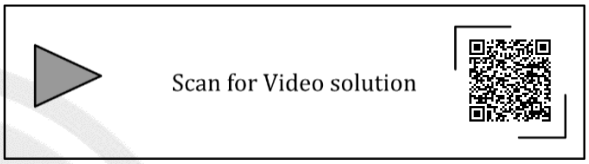
then, j = 0 to size

if i = 1
then j = 1 to size

for i = 0	for i = 1
K = 0 to 0	k = 1 to1
= 0 to 1	= 1 to 2
= 0 to 2	= 1 to 3
= 0 to 3	= 1 to size
= 0 to size	

Finding the sum of each subarray from i.e. starting from 0 to n–1 position.

Check whether the sum of elements of subarray is greater then, the maximum sum of any sub – array found till now⇒if its is then update current maximum.



4. (c)

Code – 1

```
for (i  = 0; i < n ; 1+ +)
Y [i]+ = X [0][i]
X [i] [i] = + I + j
X [0] [i]
for (i =0; I < n, i + +)
Y [i]+ = i
```

Code – 2

```
for (i = 0; i < n; i+j)
Y [i]+ = X [i] [0]
X [i] [j] = i + j
X [i] [0] = i
For (i  = 0; i < n; i ++)
Y [i] + = i;
```

Result is same in both segment

```
For (i= 0; i < n ; 1+j)
Y [i] + = X [0][i];
```

Suppose, n = 3;

i = 0 X [0] [0]

i = 1 X [0] [1]

i = 2 X [0] [2]

i = 3 X [0] [3]

In C, by default row major order is followed

⇒ elements are stored contiguously.

In Code – 2

i = 0 X [0] [0]

i = 1 X [0] [0]

i = 2 X [2] [0]

i = 3 X [3] [0]

In C, Row major order is followed

⇒ elements are not stored contiguously.

Row major order:

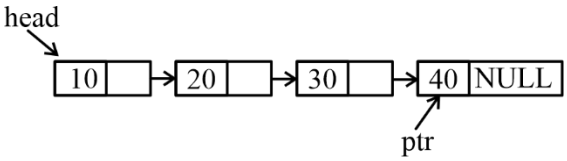
$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array}\begin{bmatrix} 0 & 1 & 2 & 3 \\ X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix}$$

Stored data row wise and access the data row wise.

Scan for Video solution

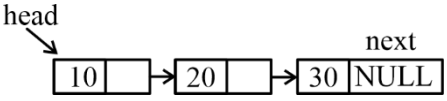
5. (d)

SLLdel:



Assume that we want to delete last node.
After Deletion we need to NULL in the second last node next field.

After deletion:

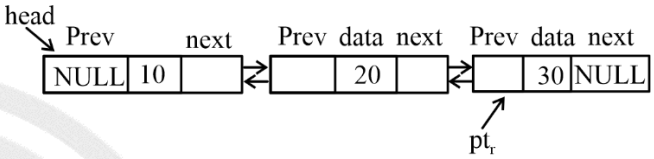


One way traversal is possible given ptr and head, how can we reach second last node ⇒ Traverse from start (head) till second last node ⇒ Traverse n – 1 nodes.

SLLDel Time complexity = O(n)

DLLdel:

- we can go to previous node as well as next node. No need to traverse from head to node to be deleted.
- In doubly linked list ptr → prev : Point to second last node



Delete operation:

ptr → Prev → next = NULL

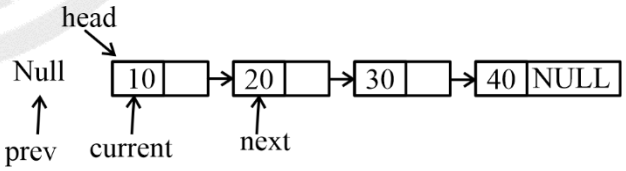
free (ptr)

It will take constant time i.e. O(1).

Scan for Video solution

6. (a)

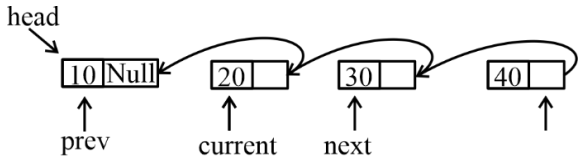
Suppose,



Program:

```
Struct node * prev, * current, * next
prev = Null;
current = head;
While (current! = NULL)
{
next = current → next;
current → next = prev;
Prev = current;
current = next;
}
```

head = prev;



TC = θ(n)

Space = θ(1)

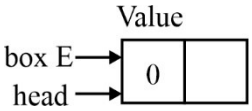
Scan for Video solution

7. (d)

Index = 0

boxE = head = (struct node*) malloc (size of (struct node))

head → value = index



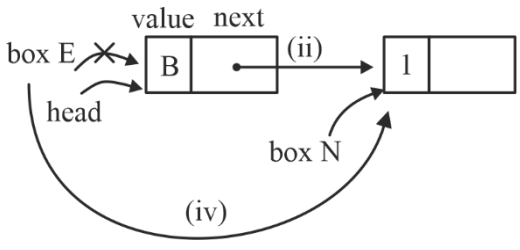
Index = 1

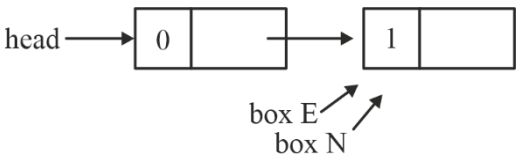
(i) box N = (struct Node*) malloc (size of (struct node))

(ii) box E → next = box N;

(iii) box N → value = index;

(iv) box E = box N





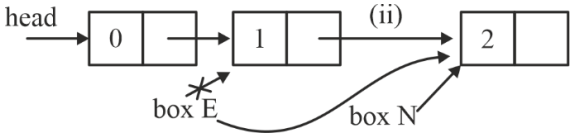
Index = 2

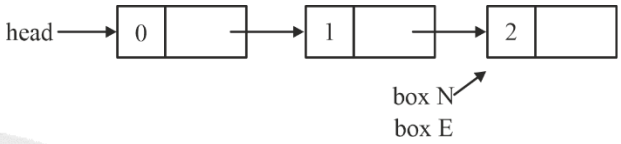
(i) box N = (struct Node*) malloc (size of (struct node))

(ii) box E → next = box N;

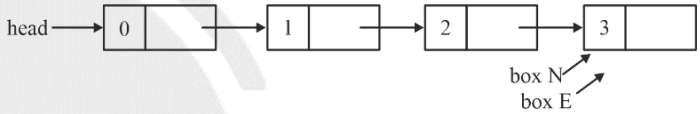
(iii) box N → value = index;

(iv) box E = box N





Same will happen for index = 3



(Last node next field contains garbage value)

Now 2nd loop starts

Index = 0

1st printf: value of index 0 is 0

head = head → next ⇒ head points to 2nd node.

2nd printf: value at Index 1 is 1

Index = 1

1st printf: value of index 1 is 1

head = head → next ⇒ head points to 3rd node.

2nd printf: value at Index 2 is 2.

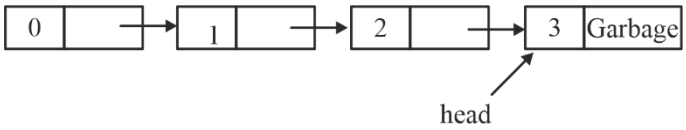
Index = 2

1st printf: value of index 2 is 2

Head = head → next ⇒ head points to last node.

2nd printf: value at Index 3 is 3.

Index = 3



1st printf: value of index 3 is 3

Head = head → next (Garbage value)

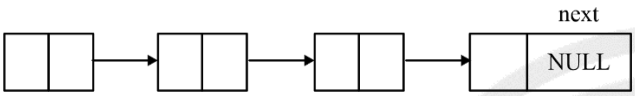
Head is pointing to some Garbage value or uniutilized memory.

2nd printf : while dereferencing head → value code may get a runtime error.

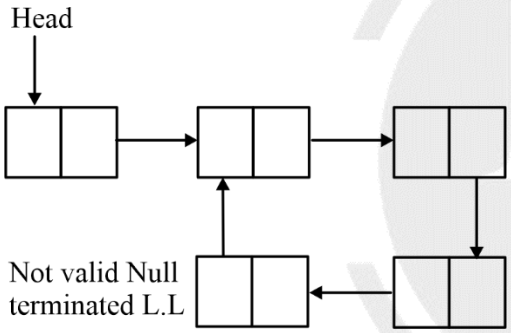
Scan for Video solution

8. (b)

Valid NULL-terminated linked list means the linked-list with last node as NULL. It can be empty or it can be non-empty

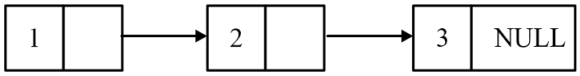


For example:

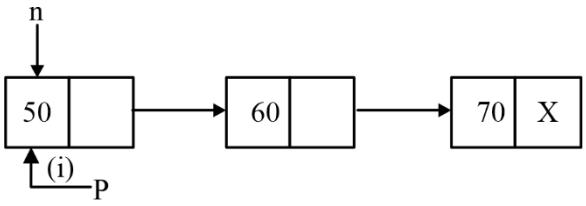


Case:-1: When both m and n are non empty linked list.

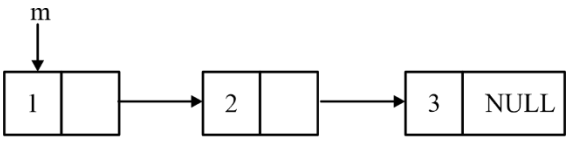
m:

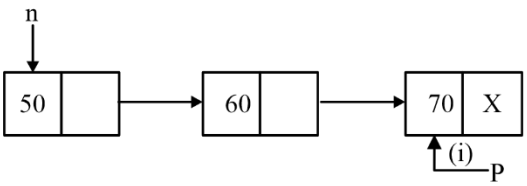


n:

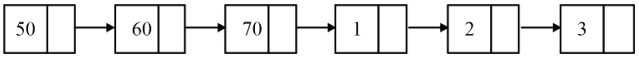


Finding last node when we came out of loop, P is pointing to last node.





After while loop:



Appended m at the end of n.

Case 2: Both m and n are null

n = NULL

m = NULL

(i) If both m and n are null, then P = NULL

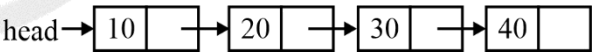
Null → Next ! = NULL

This will create an error because we are trying to dereference a null pointer. So, option (b) is the correct answer.

Scan for Video solution

9. (c)

Consdier the sorted list



If we want to insert 50 in this list

We need to traverse till end i.e, number of comparison = number of elements

i.e., to insert an element in a sorted linked list with n element n comparisons are needed, in worst case

For 1st element = 0 comparison

For 2nd element = 1 comparison

.

.

.

For nth element = (n – 1) comparison

Total comparisons needed to insert n elements

$= 0 + 1 + 2 + \ldots.(n-1)$

$= \frac{(n-1)n}{2} = O(n^2)$

Option (c) is correct.

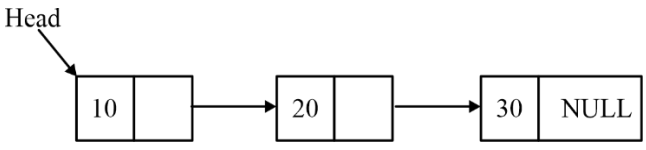
Scan for Video solution

10. (d)

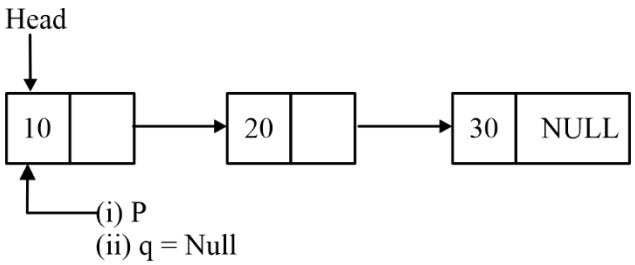
```
typedef struct node{
int value;
struct node * next;
}Node;                    // Template for node
Node * move_ to _ front (Node * head){
Node * p, *q;
if ((head = = NULL)|| (head→next = =NULL))
return head;  // Either 0 or 1 node; last node is the
first node
q = NULL;
p = head;
while (p →next ! = NULL){
q = p;
p = p → next;
}
```

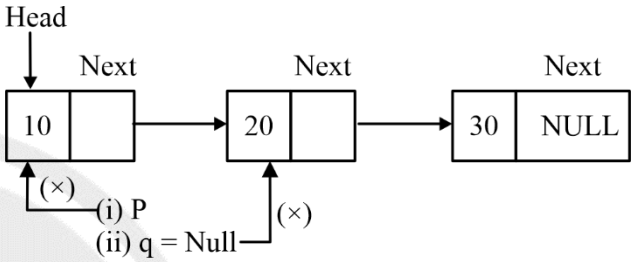
```
return head;


}
```

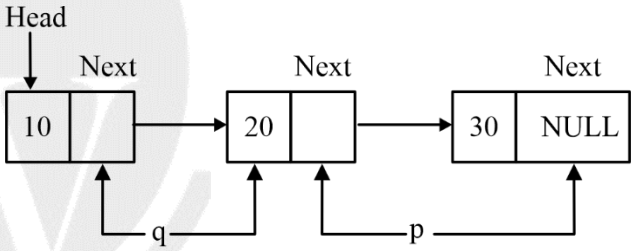


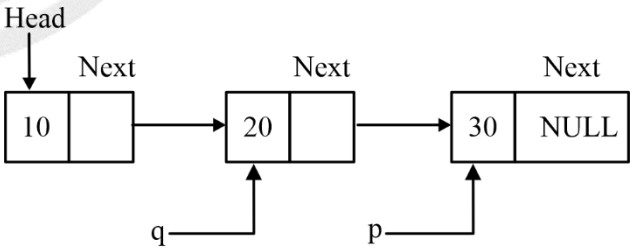
Ist iteration:



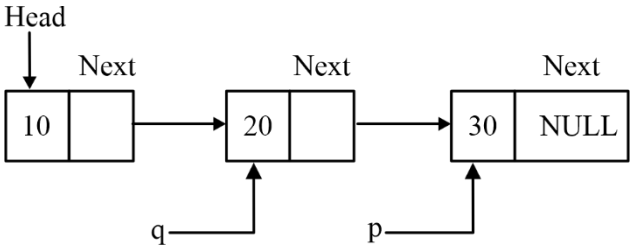


II iteration:

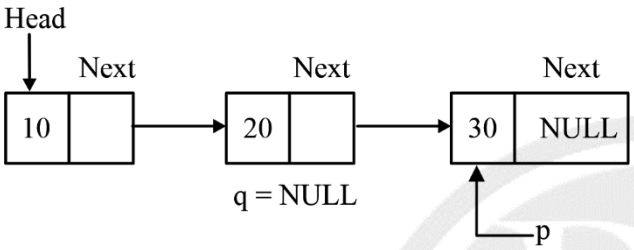


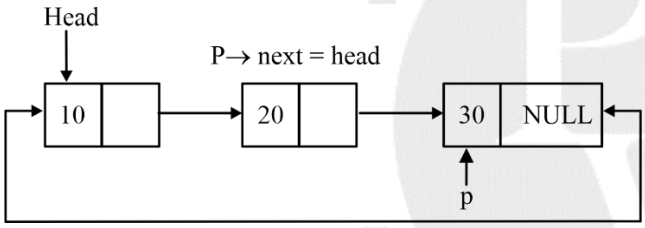


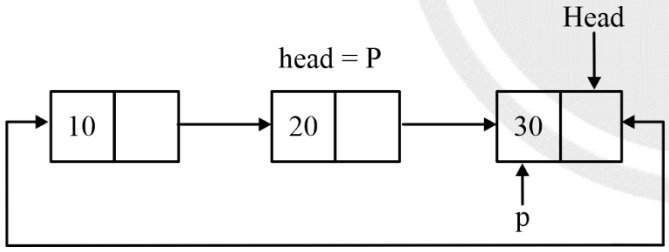
$P \rightarrow next = = Null$
loop terminate



Option (a) q = NULL ; p → next = head ; head = p;

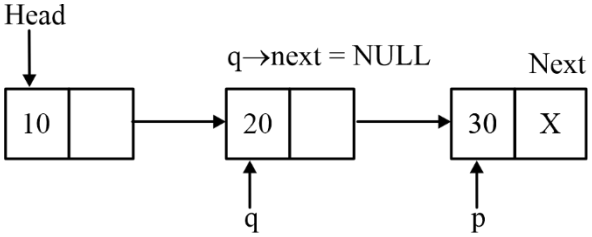


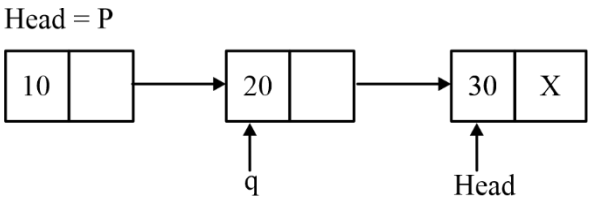


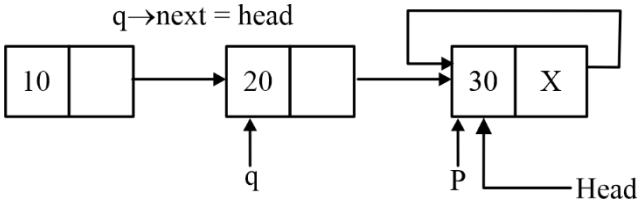


This is incorrect, this is not singly linked list. So, option (a) is wrong.

Option (b) q → next = NULL; head = p; p → next = head;

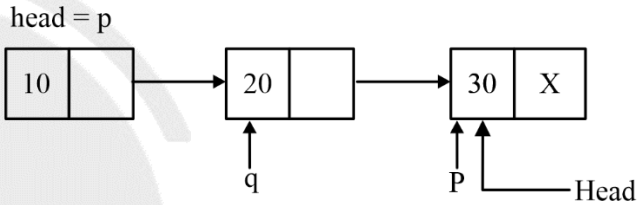




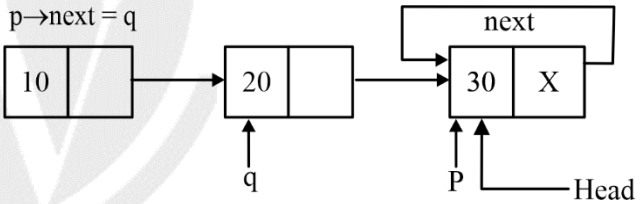


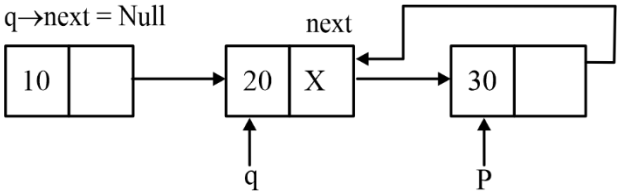
Option (b) is wrong.

Option (c) head = p; p → next = q; q → next = NULL;



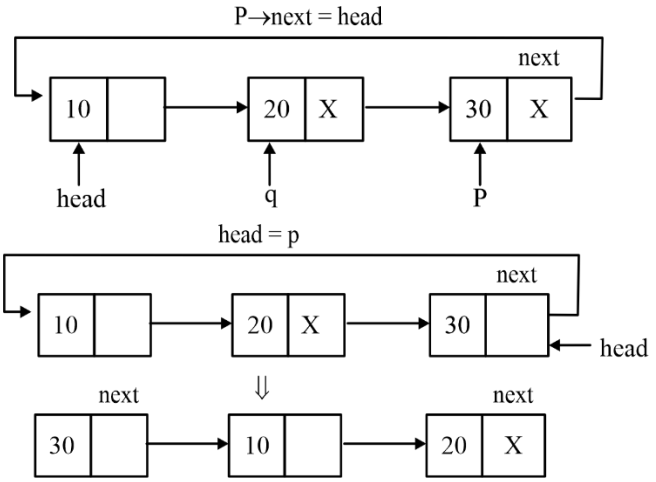
No way to reach first node





Option (c) is wrong.

Option (d) q → next = NULL; p → next = head; head = p;

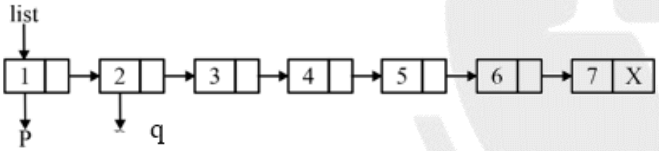


Therefore, option (d) is the correct answer.

Scan for Video solution

11. (b)

Given,

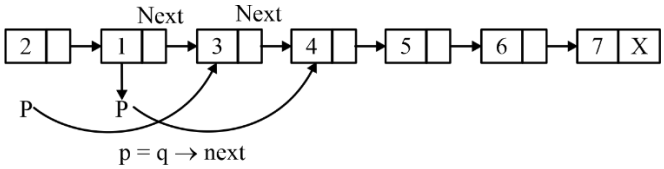


if(!list||!list→next) return; // 0 or 1 node

Swap data of node pointed by p,q

Q is valid address

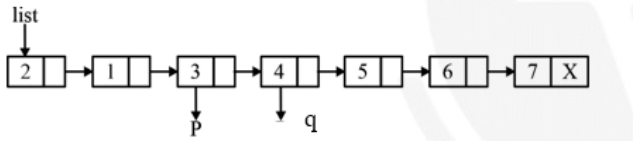
⇒ code inside loop ⇒ execute swap



p = q → next

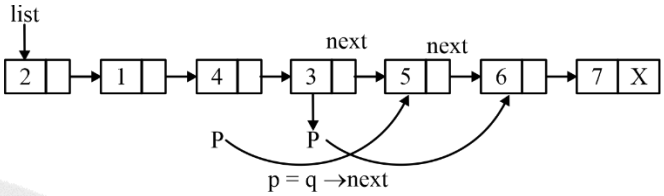
Q = p? P→ next: 0

After one iteration:



q is valid address (non-zero/true) code inside loop will execute

(i) Swap

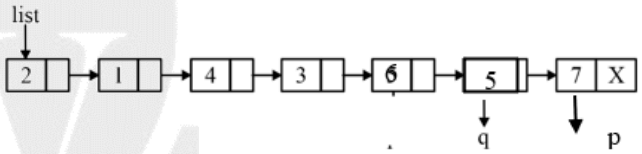


True non-zero

q = p ? P→ next : 0

⇒ p = q → next

After two iterations:



if(!list||!list→next) return;// 0 or 1 node

q is valid address code inside loop swap.

if(!list||!list→next) return;// 0 or 1 node

q = p ? p→ next: 0

q = p → next ⇒ NULL

while (q) Null ⇒ False

Loop terminates

2, 1, 4, 3, 6, 5, 7

Therefore, option (b) is the correct answer.

Scan for Video solution

❑❑❑

CHAPTER

6 STACKS AND QUEUES

Operation on Stack

1. [NAT] [GATE-2023: 2M]

Consider a sequence a of elements $a_0 = 1$, $a_1 = 5$, $a_2 = 7$, $a_3 = 8$, $a_4 = 9$, and $a_5 = 2$. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I: push the elements of a from a_0 to a_5 in that order into S.

II: enqueue the elements of a from a_0 to a_5 in that order into Q.

III: pop an element from S.

IV: dequeue an element from Q.

V: pop an element from S.

VI: dequeue an element from Q.

VII: dequeue an element from Q and push the same element into S.

VIII: Repeat operation VII three times.

IX: pop an element from S.

X: pop an element from S.

The top element of S after executing the above operations is _____.

2. [NAT] [GATE-2015: 2M]

Consider the C program below.

```
#include <stdio.h>
int * A, stkTop;
int stkFunc (int opcode, int val)
{
static int size = 0, stkTop = 0;
switch (opcode)
{
case -1: size = val; break;
case 0: if (stkTop < size)  A[stkTop++] = val; break;
default: if (stkTop) return A[– –stkTop];
}
return – 1;
}
int main ()
{
int B[20]; A = B; stkTop = – 1;
stkFunc (-1, 10);
stkFunc (0, 5);
stkFunc (0, 10);
printf("%d \ n", stkFunc(1, 0) + stkFunc (1, 0) );
}
```

The value printed by the above program is _____

Infix Postfix and Prefix Notations

3. [MCQ] [GATE-2015: 1M]

The result evaluating the postfix expression 10 5 + 60 6 / * 8 – is

(a) 284 (b) 213

(c) 142 (d) 71

Applications of Stack

4. [MCQ] [GATE-2014: 2M]

Suppose, a stack implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE (with respect to this modified stack)?

(a) A queue cannot be implemented using this stack.

(b) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes sequence of two instructions.

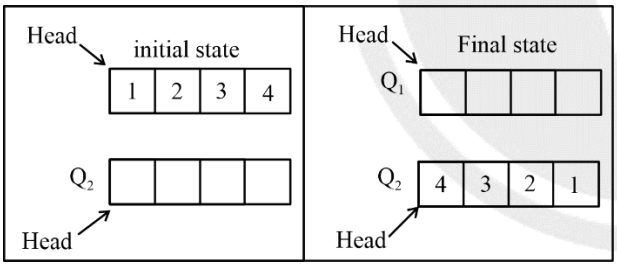
(c) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction

(d) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

Operations on Queues

5. [NAT] [GATE-2022: 2M]

Consider the queues Q_1 containing four elements and Q_2 containing none (shown as the Initial State in the figure). The only operations allowed on these two queues are Enqueue(Q, element) and Dequeue(Q). The minimum number of Enqueue operations on Q_1 required to place the elements of Q_1 in Q_2 in reverse order (shown as the Final State in the figure) without using any additional storage is_____.



6. [NAT] [GATE-2021: 1M]

Consider the following sequence of operations on an empty stack.

push(54); push(52); pop(); push(55); push(62); s = pop();

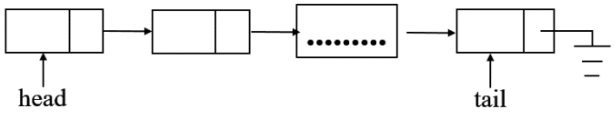
Consider the following sequence of operations on an empty queue.

enqueue(21); enqueue(24); dequeue(); enqueue(28); enqueue(32); q = dequeue();

The value of s + q is _____

7. [MCQ] [GATE-2018: 1M]

A queue is implemented using a non-circular singly linked list. The queue has a head pointer and tail pointer, as shown in the figure. Let n denote of number of nodes in the queue. Let 'enqueue' be implemented by inserting a new node at the head and 'dequeue' be implemented by deletion of a node from the tail.



Which one of the following is the time complexity of the most time-efficient implementation of enqueue and dequeue, respectively, for this data structure?

(a) $\theta(1), \theta(1)$ (b) $\theta(1), \theta(n)$

(c) $\theta(n), \theta(1)$ (d) $\theta(n), \theta(n)$

8. [MCQ] [GATE-2017: 1M]

A circular queue has been implemented using a singly linked list where each node consists of a value and a single pointer pointing to the next node. We maintain exactly two external pointers FRONT and REAR pointing to the front node and the rear node of the queue, respectively. Which of the following statements is/are CORRECT for such a circular queue, so that insertion and deletion operations can be performed in O(1) time?

I. Next pointer of front node points to the rear node.

II. Next pointer of rear node points to the front node

(a) I only (b) II only

(c) Both I and II (d) Neither I nor II

9. [MCQ] [GATE-2016: 1M]

A queue is implemented using an array such that ENQUEUE and DEQUEUE operations are performed efficiently. Which one of the following statements is CORRECT (n refers to the number of items in the queue)?

(a) Both operations can be performed in O(1) time.

(b) At most one operation can be performed in O(1) time but the worst case time for the other operation will be $\Omega(n)$.

(c) The worst case time complexity for both operations will be $\Omega(n)$.

(d) Worst case time complexity for both operations will be $\Omega(\log n)$.

10. [MCQ] [GATE-2013: 2M]

Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q){
m = k;
while (Q is not empty and m > 0) {
Dequeue (Q);
m = m – 1;
}
}
```

What is the worst case time complexity of a sequence of n MultiDequeue () operations on an initially empty queue?

(a) $\Theta(n)$ (b) $\Theta(n + k)$

(c) $\Theta(nk)$ (d) $\Theta(n^2)$

11. [MCQ] [GATE-2012: 2M]

Suppose a circular queue of capacity (n-1) elements is implemented with an array of n elements. Assume that the insertion and deletion operations are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are

(a) Full: (REAR + 1) mod n = = FRONT
Empty: REAR = = FRONT

(b) Full: (REAR + 1) mod n = = FRONT
Empty: (FRONT + 1) mod n = = REAR

(c) Full: REAR = = FRONT
Empty:(REAR + 1) mod n = = FRONT

(d) Full:(FRONT + 1) mod n = = REAR
Empty: REAR = = FRONT

Applications of Queues

12. [NAT] [GATE-2016: 2M]

Let Q denote a queue containing sixteen numbers and S be an empty stack. Head(Q) returns the element at the head of the queue Q without removing it from Q. Similarly Top(S) returns the element at the top of S without removing it from S. Consider the algorithm given below.

```
While Q is not Empty do
If S is Empty OR Top (S) ≤ Head (Q) then
X: = Dequeue (Q);
Push (S, x):
else
X: = Pop (S);
Enqueue (Q, x);
end
end
```

The maximum possible number of iterations of the while loop in the algorithm is _____.

❑❑❑

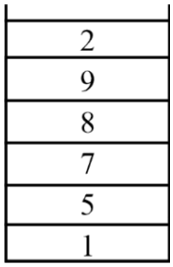
ANSWER KEY

1.	(8)	**2.**	(15)	**3.**	(c)	**4.**	(c)
5.	(0)	**6.**	(86)	**7.**	(b)	**8.**	(b)
9.	(a)	**10.**	(a)	**11.**	(a)	**12.**	(256)

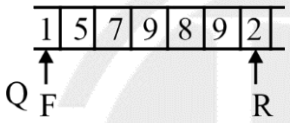
SOLUTIONS

1. (8)

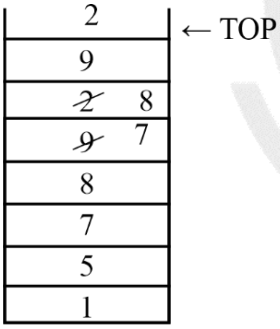
Step 1:

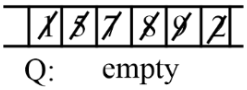


Step 2:

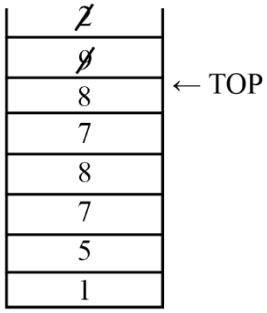


Step 3 to 8:





Step 9 & 10:



TOP element = 8

Scan for Video solution

2. (15)

stkFunc:

Implementation of stack with 3 choice/

Case 1:

Set the size of the stack

Case 0:

Push ( )

Whether stack is full or not if the stack is not full,

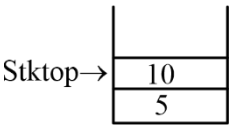
⇒ Push an element

⇒ Increment stk top

Default:

If stack is not empty then, Pop the top element & return it and decrease the top pointer.

Stack size = 10 push 5 and 10 on to stack.



stkFunc (1, 10);→ POP and return top most element 10 stkFunc (0, 5); → POP and return 5

Final output = 10 + 5 = 15.

Scan for Video solution

3. **(c)**

Two methods:

(i) without using stack

(ii) using stack

10, 5, + 60, 6, 1, *, 8, –

(i) Without Stack:

Scan from left to right & whenever an operator is encountered just put in between previous 2 operand and find result.

- 10, 5, +

10 + 5 = 15

15, 60, 6, 1, *, 8, –

(i) 15, 60, 6, /, *, 8 – put division in between 60 and 6

(ii) 15, 10, *, 8, – put * between 15 and 10

(iii) 15 * 10, 8, –

150, 8 –

(iv) 150, 8, – put – in between 150 and 8

150 – 8 = 142

Using stack:

- Scan ⇒ left to right

(i) Operand ⇒ Push it

(ii) Operator

(a) Pop top 2 elements from stack let first popped element is A & second popped elements is B.

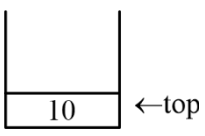
(b) Evaluate B ⊕ A (where ⊕ is the operator found)

(c) Push the result of previous step onto stack.

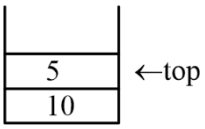
Keep repeating the above, once the input is over at the end, stack top element is the answer.

10, 5, +, 60, 6, /, *, 8, –

Push 10:



Push 5:

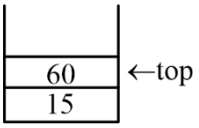


+ Operator is coming, pop the top two elements and calculating the result.

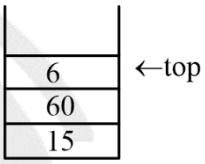
Result = 10 + 5 = 15

And push the result into stack.

Push 60:



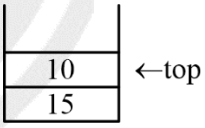
Push 6:



/ Operator is coming, pop the top two elements and calculating the result.

Result = 60 / 6 = 10. And push the result into stack.

Push 10 (result):

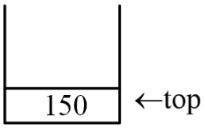


* Operator is coming, pop the top two elements and calculating the result.

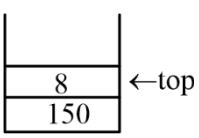
Result = 15 × 10 = 150

And push the result into stack.

Push 150 (result):



Push 8:



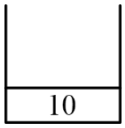
– Operator is coming, pop the top two elements and calculating the result.

Result = 150 – 8 = 142

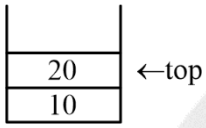
Scan for Video solution

4. (c)

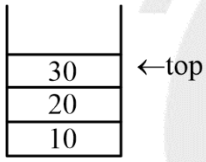
(i) Push (S, 10)



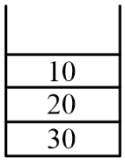
(ii) Push (S, 20)

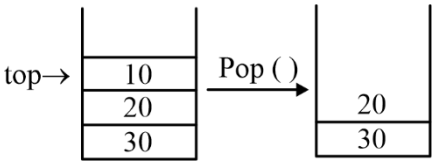


(iii) Push (S, 30)



⇓ Reverse

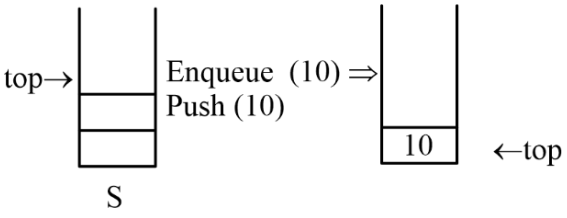




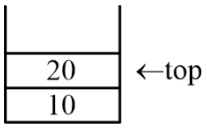
Given: Stack with 3 operation (Push/Pop/Reverse)

Queue: First in first out

(i) Let Enqueue is implemented using Push

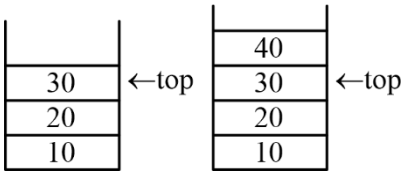


Enqueue (20) ⇒ Push (20)



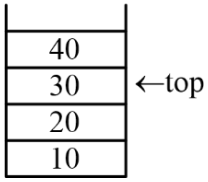
Enqueue (30) ⇒ Push (30)

Enqueue (40) ⇒ Push (40)



Dequeue?

(i) Enqueue 10
(ii) Enqueue 20
(iii) Enqueue 30
(iv) Enqueue 40



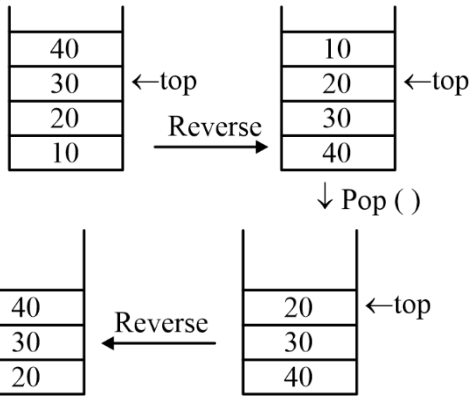
Order: $\underrightarrow{10,20,30,40}$

So, as per Queue Policy, 0 is to be deleted first ⇒ but we cannot delete any other element except top element from stack.

We need 10 at the top of stack:- **REVERSE**

Dequeue?

(i) Enqueue 10
(ii) Enqueue 20
(iii) Enqueue 30
(iv) Enqueue 40



Order: $\underrightarrow{10,20,30,40}$

(1) Dequeue → (i) Reverse, (ii) Pop, (iii) Reverse

(2) Enqueue → Push ( ) inst.

Another implement can be just reversed way

(1) Enqueue 3 ints. → (i) Reverse, (ii) Pop, (iii) Reverse → 3 ints.

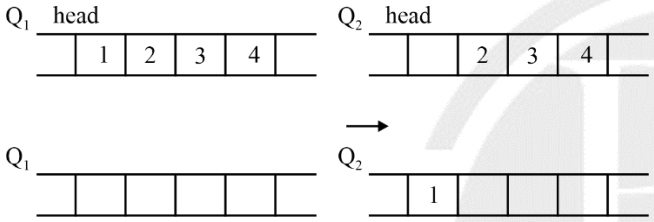
(2) Dequeue 1 ints. → Push ( ) inst. → 1 ints.

Scan for Video solution

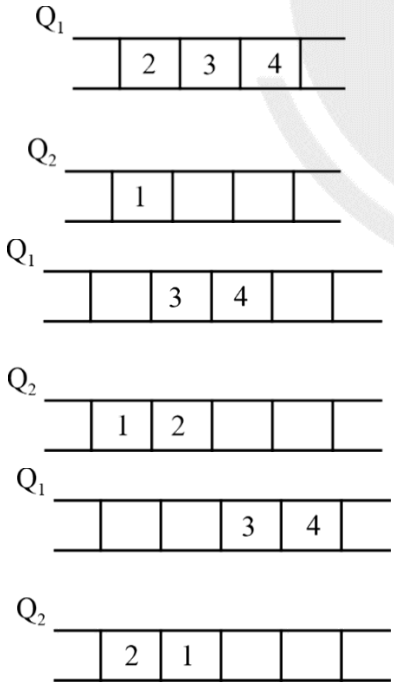
5. (0)

(i) Assume: size of $Q_2 > 4$

(ii) No addition storage

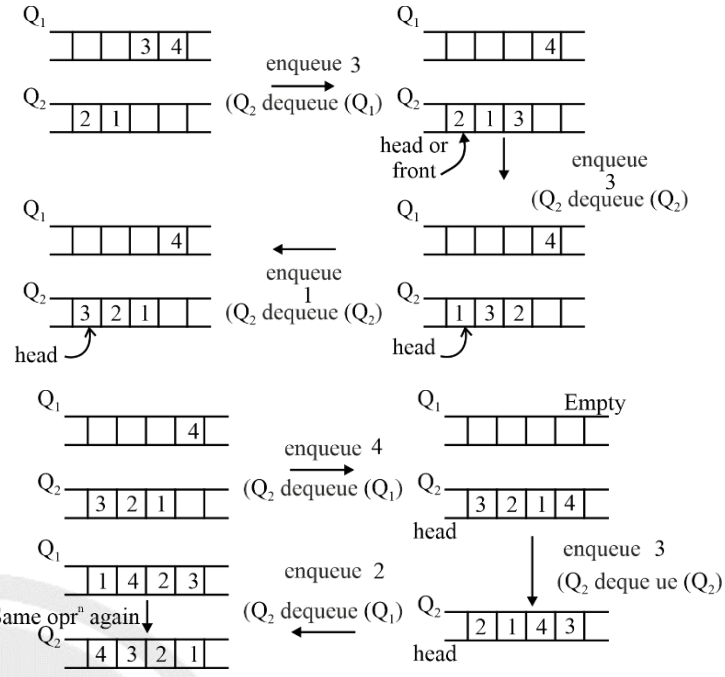


Enqueue (Q_2, Dequeue (Q_1)); No additional storage



Enque(Q_2 Deque (Q_1))

Enque(Q_2 Deque (Q_2))

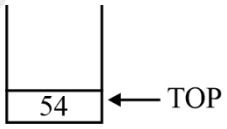


Scan for Video solution

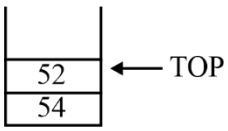
6. (86)

Stack operation:

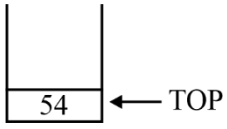
(1) Push (54):



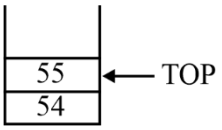
(2) Push (52):



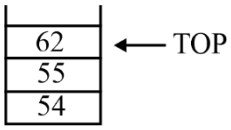
(3) POP ( ):



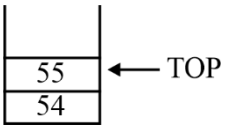
(4) Push (55):



(5) Push (62):



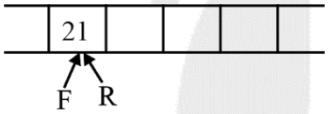
(6) s = POP ( ):



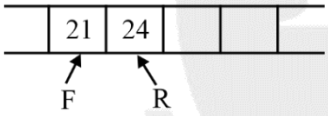
s = 62

Queue Operation:

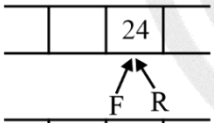
1. Enqueue(21)



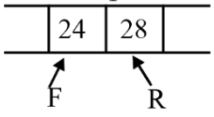
2. Enqueue(24)



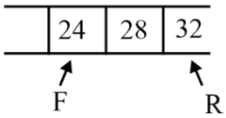
3. Dequeue



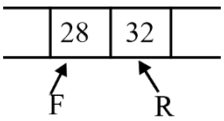
4. Enqueue(28)



5. Enqueue(28)



q = dequeue:



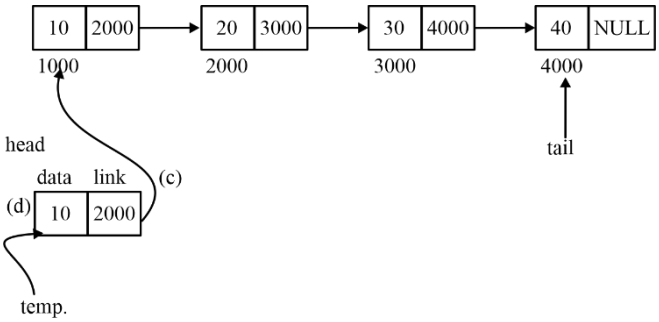
q = 24

s + q = 62 + 24 = 86

Scan for Video solution

7. (b)

struct node temp = malloc (size of (struct node))



Enqueue:

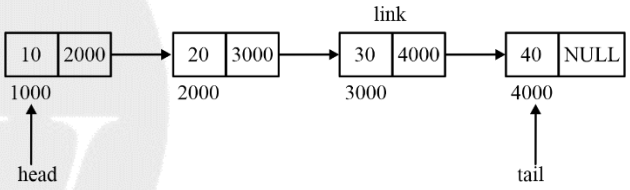
(a) struct node temp = malloc (size of (struct node))

(b) fill the data

(c) temp → link = head

(d) head = temp

fixed number of instructions ⇒ θ(1)



Dequeue:

(1) After deletion of last node second last node will become the last node & that's why we must assign NULL in the link field of 2nd last node.

How to delete last node?

We need to reach second last node or to delete a node you need to travel from 1st node to last node.

Because we have singly limited list

⇒ Traversal till last node is depends on the number of nodes.

Time Complexity = $\Theta(n)$

Scan for Video solution

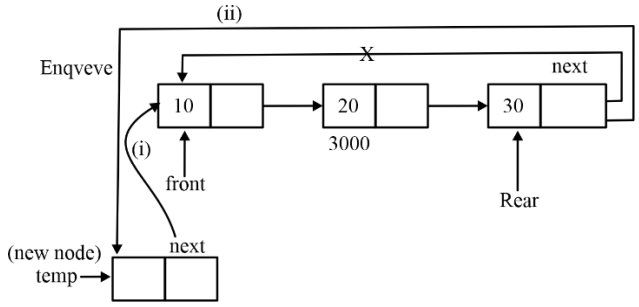
8. (b)

Enqueue → O (1)

Dequeue → O (1)

Rear → next points to front

Enqueue:

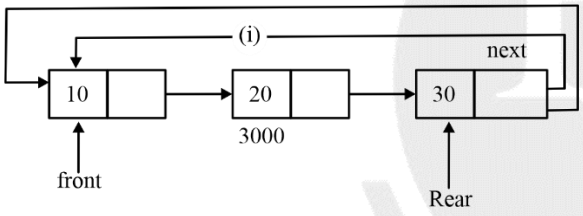


(i) temp → next = front or temp → next = Rear → next;

(ii) Rear → next = temp;

(iii) Rear = temp;

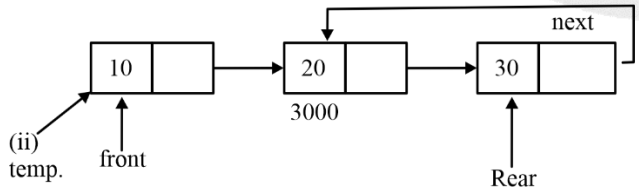
Dequeue:



Delete node 10.

(i) Rear → next (always point to front)

Rear → next = front → next



(ii) temp = front

(iii) front = Rear → next

(iv) free (temp)

Hence, option (b) is correct.

Scan for Video solution

9. (a)

Enqueue:

- If the array is full no need to do anything.
- Else Insert the new element at the end (Rear end) & also increment rear value.

Dequeue:

If the array is empty then stop else delete element from front end & also update the value of front.

Both can be perform in O(1) time.

Scan for Video solution

10. (a)

(1) **Enqueue:** Insert an element

(2) **Dequeue:** Delete an element

3 Operations:

(1) Dequeue

(2) Enqueue

(3) Multi Dequeue

Because the queue is initially empty

⇒ Number of dequeue ≯ Number of Enqueue total number of operation = θ(n)

Scan for Video solution

11. (a)

Condition for Empty: Rear = Front = 0

(b) According to option b, Empty: (FRONT + 1) mod n = = REAR

n = 5 size

(Front +1) mode 5 = = 0

(0+1) mode 5 = = 0

1 = = 0

False, That's why option b is wrong.

Condition for FULL: (REAR +1) mod n == Front

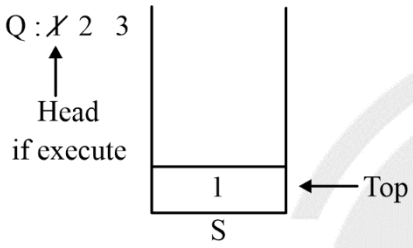
Therefore, option (a) is the correct answer.

Scan for Video solution

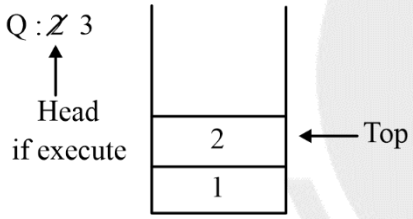
12. (256)

Let reduce the number of element from 16 to 3 to solve the questions in easy way.

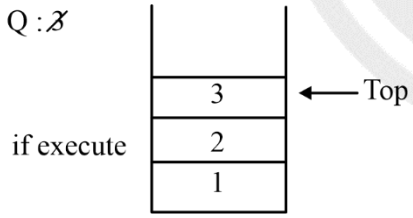
(i) Q : ✗ 2 3



(ii) Q : ✓ 3



(iii) Q : ✗

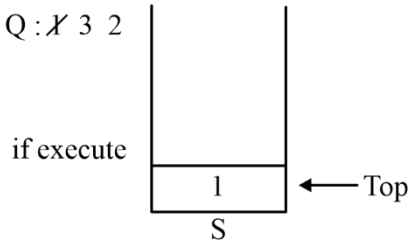


We can say

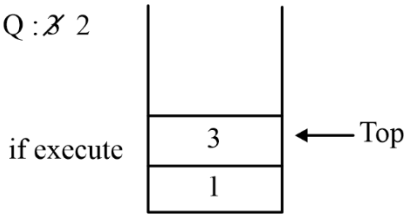
(i) Q: 1,2,3 → 3 iterations

Now let's take

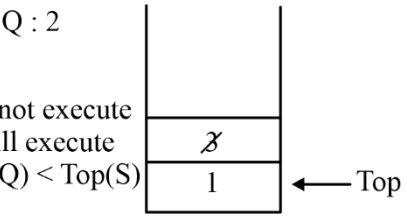
(i) Q : ✗ 3 2



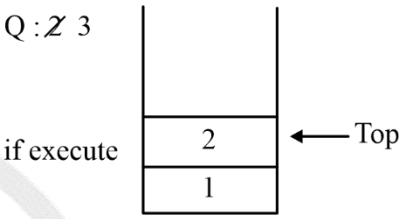
(ii) Q : ✗ 2



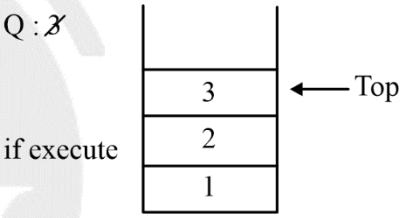
(iii) Q : 2



(iv) Q : ✓ 3



(v) Q : ✗

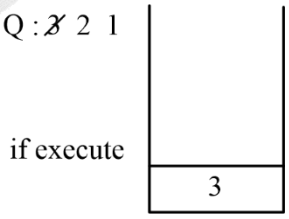


We can say for

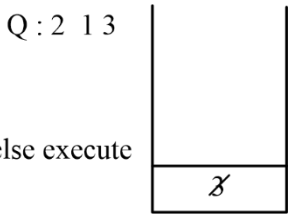
Q: 1, 3, 2 → 5 iterations

Now let's take

(i) Q : ✗ 2 1

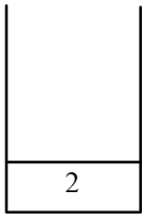


(ii) Q : 2 1 3



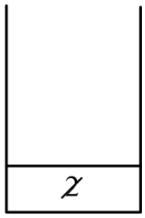
In these 2-operation 3 pushed to stack and then popped and go back to queue.

(iii) Q : 2 1 3



if execute

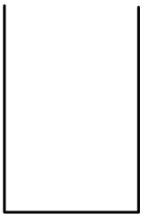
(iv) Q : 1 3 2



else execute

In these 2-iteration 2 pushed and then popped and go back to queue.

(v) Q : 1 3 2



Already know it will take 5 more iteration.

Total iteration = 4 + 5 = 9 $\Rightarrow$ (3×3)

So, for 16 element it will take

16 × 16 = 256 operation.

Scan for Video solution

❑❑❑

CHAPTER

7

TREES

Implementation of Tree

1. [MCQ] [GATE-2023 2M]

Let A be a priority queue for maintaining a set of elements. Suppose *A* is implemented using a max-heap data structure. The operation EXTRACT-MAX(A) extracts and deletes the maximum element from *A*. The operation INSERT(A, key) inserts a new element *key* in *A*. The properties of a max-heap are preserved at the end of each of these operations. When A contains n elements, which one of the following statements about the worst case running time of these two operations is TRUE?

(a) Both EXTRACT-MAX(A) and INSERT(A, key) run in $O(1)$.

(b) Both EXTRACT-MAX(A) and INSERT(*A*, key) run in $O(\log(n))$.

(c) EXTRACT-MAX(*A*) runs in $O(1)$ whereas INSERT(*A*, *key*) runs in $O(n)$.

(d) EXTRACT-MAX(*A*) runs in $O(1)$ whereas INSERT(A, key) runs in $O(\log(n))$.

2. [MCQ] [GATE-2023: 1M]

Which one of the following sequences when stored in an array at locations

A[1], . . . , A[10] forms a max-heap?

(a) 23, 17, 10, 6, 13, 14, 1, 5, 7, 12

(b) 23, 17, 14, 7, 13, 10, 1 ,5, 6, 12

(c) 23, 17, 14, 6, 13, 10, 1, 5, 7, 15

(d) 23, 14, 17, 1, 10, 13, 16, 12, 7, 5

3. [NAT] [GATE-2020: 2M]

Consider the array representation of a binary min-heap containing 1023 elements. The minimum number of comparisons required to find the maximum in the heap is_____.

4. [MCQ] [GATE-2019: 2M]

Consider the following statements:

I. The smallest element in a max-heap is always at a leaf node

II. The second largest element in a max-heap is always a child of a root node

III. A max-heap can be constructed from a binary search tree in $\theta(n)$ time

IV. A binary search tree can be constructed from a max-heap in $\theta(n)$ time

Which of the above statements are TRUE?

(a) I, II and III (b) I, II and IV

(c) I, III and IV (d) II, III and IV

5. [MCQ] [GATE-2021: 1M]

Let H be a binary min-heap consisting of n elements implemented as an array. What is the worst case time complexity of an optimal algorithm to find the maximum element in H?

(a) $\Theta(1)$ (b) $\Theta(\log n)$

(c) $\Theta(n)$ (d) $\Theta(n \log n)$

6. [MCQ] [GATE-2017: 1M]

Consider the following array of elements $\langle 89,19,50,17,12,15,2,5,7,11,6,9,100\rangle$.

The minimum number of interchanges needed to convert it into a max-heap is

(a) 4 (b) 5

(c) 2 (d) 3

7. [NAT] [GATE-2016: 2M]

A complete binary min-heap is made by including each integer in [1,1023] exactly once. The depth of a node in the heap is the length of the path from the root of the heap to that node. Thus, the root is at depth 0. The maximum depth at which integer 9 can appear is _____.

8. [MCQ] [GATE-2016: 2M]

An operator delete(i) for a binary heap data structure is to be designed to delete the item in the i-th node. Assume that the heap is implemented in an array and i refers to the i-th index of the array. If the heap tree has depth d (number of edges on the path from the root to the farthest leaf), then what is the time complexity to re-fix the heap efficiently after the removal of the element?

(a) O(1) (b) O(d) but not O(1)
(c) $O(2^d)$ but not O(d) (d) $O(d2^d)$ but not $O(2^d)$

9. [MCQ] [GATE-2014: 1M]

Consider the C function given below. Assume that the array list A contains n (> 0) elements, sorted in ascending order.

```
int ProcessArray (int *list A, int x, int n){
int i, j, k;
i = 0;
j = n − 1;
do {
k = ( i + j ) / 2;
if ( x < = listA[k] )
j = k − 1;
If ( listA [k] < = x )
i = k + 1;
}
while ( i < = j );
if( listA[k] = = x ) return ( k );
else
return – 1;}
```

Which one of the following statements about the function Process Array is CORRECT?

(a) It will run into an infinite loop when x is not in list A.
(b) It is an implementation of binary search.
(c) It will always find the maximum element in list A
(d) It will return – 1 even when x is present in list A.

10. [MCQ] [GATE-2014: 2M]

Consider the pseudocode given below. The function **DoSomething ( )** takes as a argument a pointer to the root of an arbitrary tree represented by the **leftMostChild - rightSibling** representation. Each node of the tree is of type treeNode.

```
typedef struct treeNode* treeptr;
struct treeNode
{
treeptr leftMostChild, rightSibling;
};
int DoSomething (treeptr tree)
{
int value = 0;
if (tree ! = NULL) {
if (tree → left MostChild = = NULL) value = 1;
else
value = DoSomething(tree → leftMostChild);
value = value + DoSomething(tree → rightSibling);
}
return (value);
}
```

When the pointer to the root of a tree is passed as the argument to **DoSomething,** the value returned by thye function corresponds to the

(a) number of internal nodes in the tree.
(b) height of the tree.
(c) number of nodes without a right sibling in the tree.
(d) number of leaf nodes in the tree.

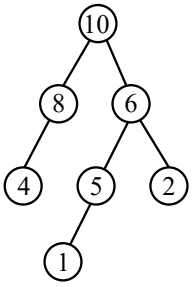
11. [MCQ] [GATE-2014: 1M]

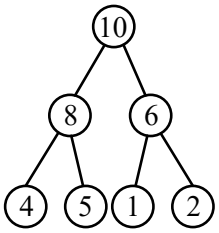
A priority queue is implemented as Max-Heap. Initially, it has **5** elements. The level-order traversal of the heap is: **10, 8, 5, 3, 2**. Two new elements **1** and **7** are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

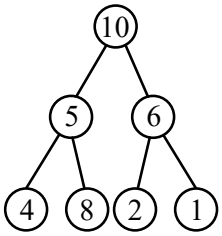
(a) 10, 8, 7, 3, 2, 1, 5 (b) 10, 8, 7, 2, 3, 1, 5
(c) 10, 8, 7, 1, 2, 3, 5 (d) 10, 8, 7, 5, 3, 2, 1

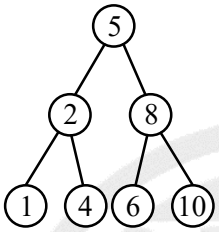
12. [MCQ] [GATE-2011: 1M]

A max-heap is a heap where the value of each parent is greater than or equal to the value of its children. Which of the following is a max-heap?

(a) 

(b) 

(c) 

(d) 

Statements for linked answer for next two question(13 & 14)

13. [MCQ] [GATE-2009: 2M]

Which one of the following array represents a binary max-heap?

(a) {25, 12, 16, 13, 10, 8, 14}
(b) {25, 14, 13, 16, 10, 8, 12}
(c) {25, 14, 16, 13, 10, 8, 12}
(d) {25, 14, 12, 13, 10, 8, 16}

14. [MCQ] [GATE-2009: 2M]

Consider a binary max-heap implemented using an array.

What is the content of the array after two delete operations on the correct answer to the previous question?

(a) { 14, 13, 12, 10, 8 } (b) { 14, 12, 13, 8, 10 }
(c) { 14, 13, 8, 12, 10 } (d) { 14, 13, 12, 8, 10 }

15. [MCQ] [GATE-2008: 1M]

We have a binary heap on **n** elements and wish to insert **n** more elements (not necessarily one after another) into this heap. The total time required for this is:

(a) $\Theta(log\ n)$ (b) $\Theta(n)$
(c) $\Theta(n\ log\ n)$ (d) $\Theta(n^2)$

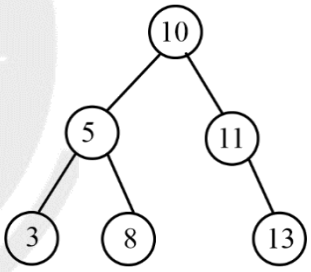
Tree Traversal

16. [MCQ] [GATE-2023: 2M]

Consider the C function foo and the binary tree shown.

```
typedef struct node {
    int val;
   struct node *left, *right;
} node;
int foo( node *p ) {
  int retval;
  if ( p = = NULL )
     return 0;
  else {
  retval = p → val + foo( p  → left ) + foo( p →right
        );
     printf( "%d ", retval );
     return retval;
  }
}
```

When foo is called with a pointer to the root node of the given binary tree, what will it print?



(a) 3 8 5 13 11 10 (b) 3 5 8 10 11 13
(c) 3 8 16 13 24 50 (d) 3 16 8 50 24 13

17. [MCQ] [GATE-2021: 1M]

Consider the following statements.

S_1: The sequence of procedure calls corresponds to a preorder traversal of the activation tree.

S_2: The sequence of procedure returns corresponds to a postorder traversal of the activation tree.

Which one of the following options is correct?

(a) S_1 is true and S_2 is false
(b) S_1 is false and S_2 is true
(c) S_1 is true and S_2 is true
(d) S_1 is false and S_2 is false

18. **[MCQ]** **[GATE-2020: 1M]**

The preorder traversal of a binary search tree is 15 10, 12, 11, 20, 18, 16, 19.

Which one of the following is the postorder traversal of the tree?

(a) 20, 19, 18, 16, 15, 12, 11, 10
(b) 11, 12, 10, 16, 19, 18, 20, 15
(c) 19, 16, 18, 20, 11, 12, 10, 15
(d) 10, 11, 12, 15, 16, 18, 19, 20

19. **[NAT]** **[GATE-2018: 1M]**

The postorder traversal of a binary tree is 8,9,6,7,4,5,2,3,1. The inorder traversal of the same tree is 8,6,9,4,7,2,5,1,3. The height of a tree is the length of the longest path from the root to any leaf. The height of the binary tree above is _____

20. **[MCQ]** **[GATE-2017: 2M]**

The pre-order traversal of a binary search tree is given by 12,8,6,2,7,9,10,16,15,19,17,20. Then the post-order traversal of this tree is:

(a) 2,6,7,8,9,10,12,15,16,17,19,20
(b) 2,7,6,10,9,8,15,17,20,19,16,12
(c) 7,2,6,8,9,10,20,17,19,15,16,12
(d) 7,6,2,10,9,8,15,16,17,20,19,12

21. **[MCQ]** **[GATE-2015: 1M]**

Which of the following is/are correct inorder traversal sequence(s) of binary search tree(s)?

1. 3, 5, 7, 8, 15, 19, 25
2. 5, 8, 9, 12, 10, 15, 25
3. 2, 7, 10, 8, 14, 16, 20
4. 4, 6, 7, 9, 18, 20, 25

(a) 1 and 4 only (b) 2 and 3 only
(c) 2 and 4 only (d) 2 only

22. **[MCQ]** **[GATE-2013: 2M]**

The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree?

(a) 10, 20, 15, 23, 25, 35, 42, 39, 30
(b) 15, 10, 25, 23, 20, 42, 35, 39, 30
(c) 15, 20, 10, 23, 25, 42, 35, 39, 30
(d) 15, 10, 23, 25, 20, 35, 42, 39, 30

23. **[MCQ]** **[GATE-2008: 2M]**

The following three are known to be the preorder, inorder and postorder sequences of a binary tree. But it is not known which is which.

I. **MBCAFHPYK**
II. **KAMCBYPFH**
III. **MABCKYFPH**

Pick the true statement from the following.

(a) I and II are preorder and inorder sequences, respectively
(b) I and III are preorder and postorder sequences, respectively
(c) II is the inorder sequence, but nothing more can be said about the other two sequences
(d) II and III are the preorder and inorder sequences, respectively

Binary Search Trees

24. **[NAT]** **[GATE-2022: 1M]**

Suppose a binary search tree with 1000 distinct elements is also a complete binary tree. The tree is stored using the array representation of binary heap trees. Assuming that the array indices start with 0, the 3rd largest element of the tree is stored at index_____.

25. **[MCQ]** **[GATE-2021: 1M]**

A binary search tree T contains n distinct elements. What is the time complexity of picking an element in T that is smaller than the maximum element in T?

(a) θ (n log n) (b) θ (n)
(c) θ (log n) (d) θ (1)

26. **[MCQ]** **[GATE-2017: 1M]**

Let T be a binary search tree with 15 nodes.

The minimum and maximum possible heights of T are:

Note: The height of a tree with a single node is 0.

(a) 4 and 15 respectively (b) 3 and 14 respectively
(c) 4 and 14 respectively (d) 3 and 15 respectively

27. [NAT] [GATE-2016: 2M]

The number of ways in which the numbers 1,2,3,4,5,6,7 can be inserted in an empty binary search tree, such that the resulting tree has height 6, is _____.

Note: The height of a tree with a single node is 0.

28. [MCQ] [GATE-2015: 1M]

While inserting the elements 71,65,84,69,67,83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is

(a) 65 (b) 67

(c) 69 (d) 83

29. [NAT] [GATE-2015: 1M]

A binary tree T has 20 leaves. The number of nodes in T having two children is _____

30. [MCQ] [GATE-2015: 1M]

Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4.

Array Index	1	2	3	4	5	6	7	8	9
Value	40	30	20	10	15	16	17	8	4

Now consider that a value 35 is inserted into this heap. After insertion, the new heap is

(a) 40, 30, 20, 10, 15, 16, 17, 8, 4, 35

(b) 40, 35, 20, 10, 30, 16, 17, 8, 4, 15

(c) 40, 30, 20, 10, 35, 16, 17, 8, 4, 15

(d) 40, 35, 20, 10, 15, 16, 17, 8, 4, 30

31. [MCQ] [GATE-2015: 1M]

The height of a tree is the length of the longest root-to-leaf path in it.

The maximum and minimum number of nodes in a binary tree of height 5 are

(a) 63 and 6, respectively

(b) 64 and 5, respectively

(c) 32 and 6, respectively

(d) 31 and 5, respectively

32. [NAT] [GATE-2014: 1M]

Consider a rooted n node binary tree represented using pointers. The best upper bound on the time required to determine the number of subtrees having exactly 4 nodes is $O(n^a \log^b n)$. Then the value of a + 10b is____.

33. [MCQ] [GATE-2012: 2M]

The height of a tree is defined as the number of edges on the longest path in the tree. The function shown in the pseudo-code below is invoked as the height (root) to compute the height of a binary tree rooted at the tree pointer root.

```
int height (treeptr n)
 {
 if ( n = = NULL) return –1;
 if ( n → left = = null )
 if ( n → right = = NULL ) return 0;
    else return  B1 ;//Box1
 else {
    h1 = height (n → left);
    if (n → right = = NULL ) return (1 + h1);
   else {
    h2 = height ( n → right );
    return  B2 ;//Box2
   }
 }
}
```

The appropriate expression for the two boxes B1 and B2 are

(a) B1: (1 + height (n → right))
B2: (1 + max (h1, h2))

(b) B1: (height (n → right))
B2: (1 + max (h1, h2))

(c) B1: height (n → right))
B2: max (h1, h2)

(d) B1: (1 + height (n → right))
B2: max (h1, h2))

34. [MCQ] [GATE-2011: 2M]

We are given a set of **n** distinct elements and an unlabeled binary tree with **n** nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree?

(a) 0 (b) 1

(c) n! (d) $\frac{1}{(n+1)} {}^{2n}C_n$

35. [MCQ] [GATE-2008: 2M]

You are given the postorder traversal, **P**, of a binary search tree on the **n** elements 1, 2, …., n. You have to determine the unique binary search tree that has P as its postorder traversal. What is the time complexity of the most efficient algorithm for doing this?

(a) $\Theta(log\ n)$

(b) $\Theta(n)$

(c) $\Theta(nlog\ n)$

(d) none of the above, as the tree cannot be uniquely determined.

36. [MCQ] [GATE-2008: 2M]

A binary tree with n > 1 nodes has n_1, n_2 *and* n_3 nodes of degree one, two and three respectively. The degree of a node is a defined as the number of its neighbours.

Starting with the above tree, while there remains a node v of degree two in the tree, add an edge between the two neighbors of v and then remove v from the tree.

How many edges will remain at the end of the process?

(a) $2 * n_1 - 3$ (b) $n_2 + 2 * n_1 - 2$

(c) $n_3 - n_2$ (d) $n_2 + n_1 - 2$

37. [MCQ] [GATE-2008: 2M]

A binary tree with n > 1 nodes has n_1, n_2 *and* n_3 nodes of degree one, two and three respectively. The degree of a node is a defined as the number of its neighbours.

n_3 can be expressed as:

(a) $n_1 + n_2 - 1$ (b) $n_1 - 2$

(c) $[((n_1+n_2)/2)]$ (d) $n_2 - 1$

38. [MCQ] [GATE-2008: 2M]

A Binary Search Tree (BST) stores values in the range 37 to 573. Consider the following sequence of keys.

I. 81, 537, 102, 439, 285, 376, 305

II. 52, 97, 121, 195, 242, 381, 472

III. 142, 248, 520, 386, 345, 270, 307

IV. 550, 149, 507, 395, 463, 402, 270

Suppose the BST has been unsuccessfully searched for key **273**. Which all of the above sequence list nodes in the order in which we could have encountered them in the search?

(a) II and III only (b) I and III only

(c) III and IV only (d) III only

AVL Trees

39. [MCQ] [GATE-2020: 1M]

What is the worst case time complexity of inserting n^2 elements into an AVL-tree with n elements initially?

(a) $\Theta(n^4)$ (b) $\Theta(n^2)$

(c) $\Theta(n^2 \log n)$ (d) $\Theta(n^3)$

40. [MCQ] [GATE-2009: 2M]

What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

(a) 2 (b) 3

(c) 4 (d) 5

41. [MCQ] [GATE-2008: 1M]

Which of the following is TRUE?

(a) The cost of searching an AVL tree is $\Theta(\log n)$ but that of a binary search tree is O(n)

(b) The cost of searching an AVL tree is $\Theta(\log n)$ but that of a complete binary tree is $\Theta(n \log n)$

(c) The cost of searching a binary search tree is O (log n) but that of an AVL tree is $\Theta(n)$

(d) The cost of searching an AVL tree is $\Theta(n \log n)$ but that of a binary search tree is O(n)

❑❑❑

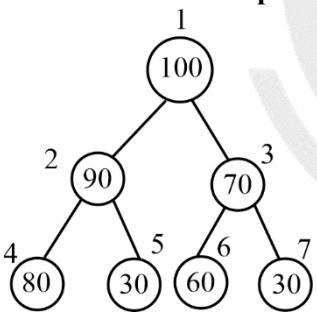
ANSWER KEY

1.	(b)	**2.**	(b)	**3.**	(511 to 511)	**4.**	(a)
5.	(c)	**6.**	(d)	**7.**	(8 to 8)	**8.**	(b)
9.	(b)	**10.**	(d)	**11.**	(a)	**12.**	(b)
13.	(c)	**14.**	(d)	**15.**	(b)	**16.**	(c)
17.	(c)	**18.**	(b)	**19.**	(4 to 4)	**20.**	(b)
21.	(a)	**22.**	(d)	**23.**	(d)	**24.**	(509 to 509)
25.	(d)	**26.**	(b)	**27.**	(64 to 64)	**28.**	(b)
29.	(19 to 19)	**30.**	(b)	**31.**	(a)	**32.**	(1 to 1)
33.	(a)	**34.**	(b)	**35.**	(b)	**36.**	(a)
37.	(b)	**38.**	(d)	**39.**	(c)	**40.**	(b)
41.	(a)						

SOLUTIONS

1. (b)

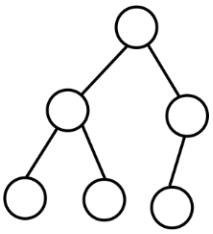
Suppopse we have a max heap



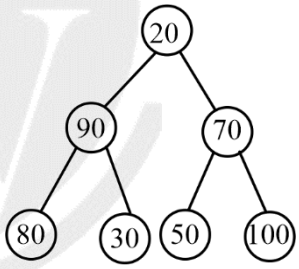
Array Representation:

1	2	3	4	5	6	7
100	90	70	80	30	50	20

Structure after delete an element:



Swap last with first element:



[Now Heapify]

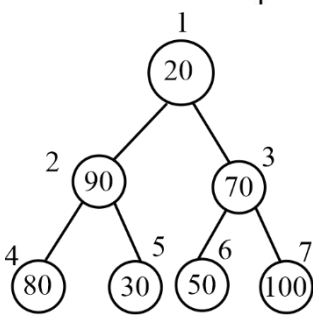
Operations:

A[1]: Max element

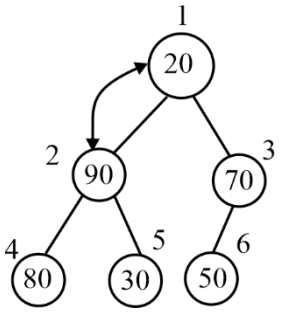
Swap A[1] → A[n]

n = n – 1

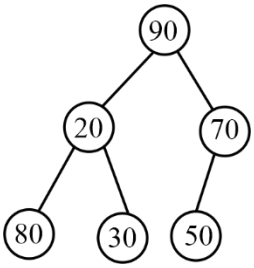
1	2	3	4	5	6	7
20	90	70	80	30	50	100



Delete element 100:

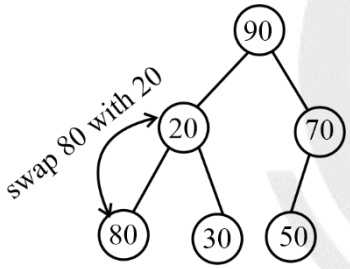


Swap 90 with 20:



Apply heapify method:

Heapify method will take O(n) time.

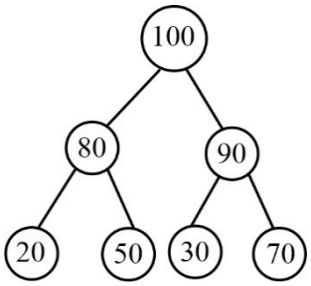


Height of heap = O(logn)

EXTRACT Max = O(logn)time

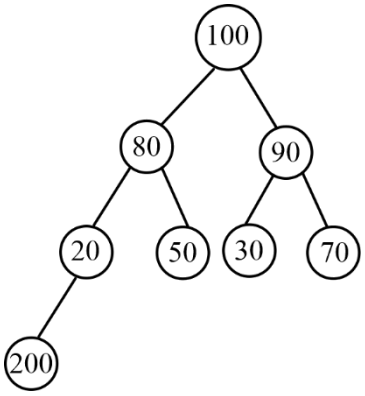
INSERT:

Suppose a max heap

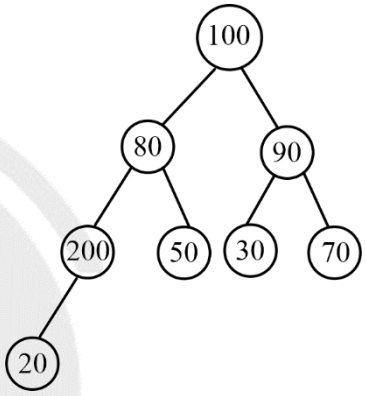


Insert 200 in given heap

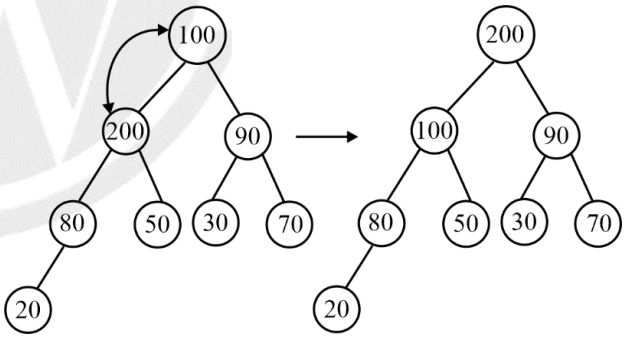
Insert (A,key) ⇒ O(logn)



Parent is smallelr (swap)



200 is greater than 80. So, we need to swap



Now, a max-heap

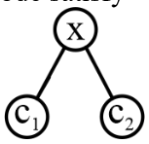
Total inertion time = O(logn)

So, operation will take O(n) time.

Scan for Video solution

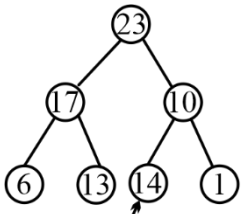
2. **(b)**

Max heap : every node satisfy



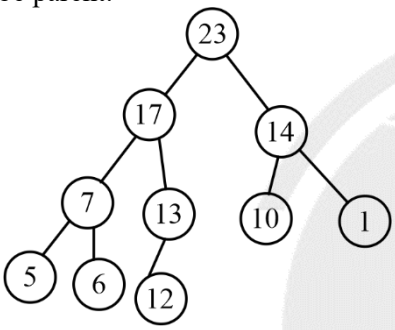
$x > (c_1, c_2)$

Node value > its child value



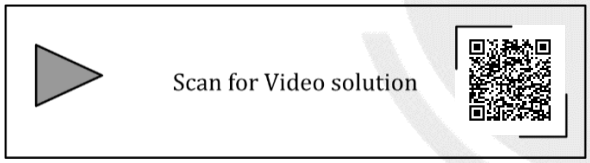
so, option (a) is wrong

14 must be parent.



It is satisfying all the property of Max- Heap.

Like that we will check option c and d.



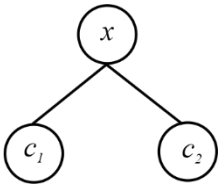
3. **(511 to 511)**

In a min-heap the maximum element can be present at a leaf node.

Property of min heap

For any node x

$\boxed{x < c_1, c_2}$



Largest among x, c_1, c_2 cannot be x.

Either of child can be maximum

$\Rightarrow$ Greatest element can be at a node which does not have any child $\Rightarrow$ leaf node.

Number of leaf node a heap with n element = $\left\lceil \frac{n}{2} \right\rceil$

In the question, n = 1023

Number of leaf nodes = $\left\lceil \frac{1023}{2} \right\rceil = 512$

To find max among 10 elements $\Rightarrow$ 9 comparisons are required

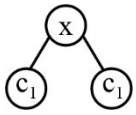
To find max among n element $\Rightarrow$ (n–1) comparisons are required

To find max among 512 element $\Rightarrow$ (512 –1) comparisons are required

= 511 comparisons.

4. **(a)**

I. First statement is true as in a max-heap, the samllest element can not be a node having some child.



$x > c_1, c_2$

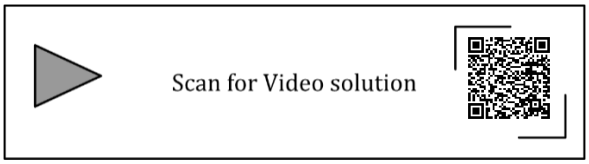
So, the smallest can be either c_1 and c_2

II. 2nd largest elements is a max-heap most be at level 1. So, this statement is also correct.

III. Third statement is true as we can use build-heap method to construct the heap in $\theta(N)$ time.

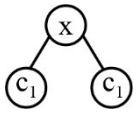
IV. is false because it will take O(n logn) time

So, (a) is correct.



5. (c)

In a min-heap, every node satisfy the property that node-value is samller than child- value.



$x < c_1, c_2$

i,e the largest among node and children can not be node i.e only child can be larger.

So, a node with some child can not be largest i.e largest element must not have any child.

$\Rightarrow$ Only leaf nodes can have largest element

Almost $\left(\frac{n}{2}\right)$ number of leaf nodes are present in a heap to find largest among these $O\left(\frac{n}{2}\right) = O(n)$ leaves possible

Number of compareisons for n elemnts $=O(n) = \Theta(n)$

Scan for Video solution

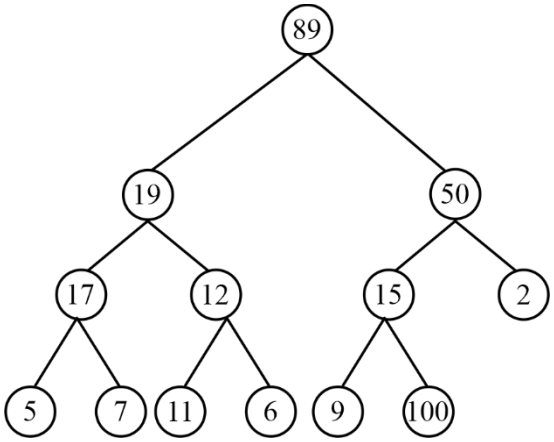
6. (d)

The given question is saying about build-heap algorithm.

Given,

89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100

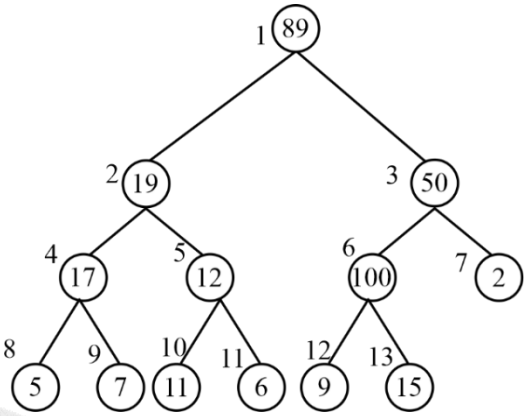
Corresponding heap:

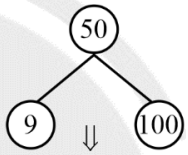


Swap

We need to work internal nodes 1,2,3,4,5,6

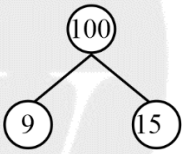
(1) with index 6

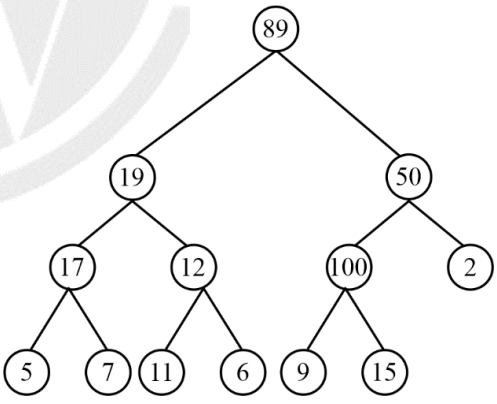


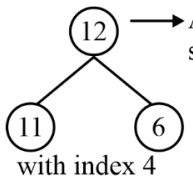


Largest among root, C_1, C_2 $\Rightarrow$ must be at root node

$\Downarrow$ 1 swap

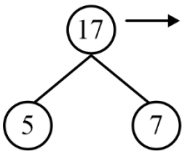






Already heap property satisfied

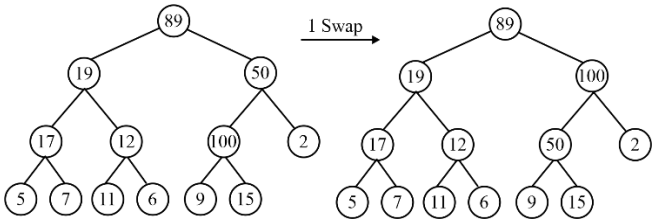
with index 4



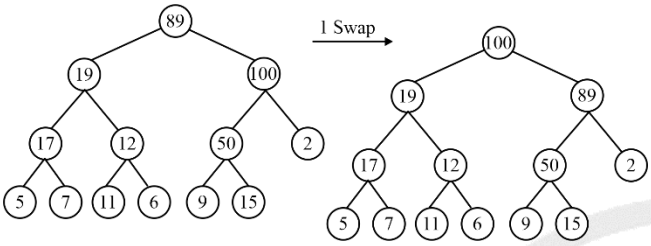
Already heap

Build- Heap

With index 3



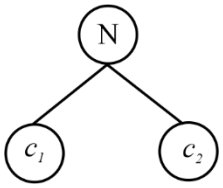
Index 2 ⇒ heap prop. Already satisfied



Total 3 swaps required. So, option (d) is the correct answer.

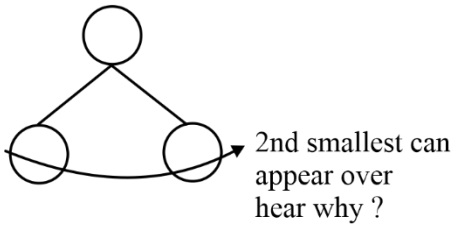
Scan for Video solution

7. (8 to 8)



$N < c_1, c_2$. In min-heap, every node is smaller than its child.

Root ⇒1st smallest



2nd smaller cannot go beyond level 1.

k^{th} smallest cannot be beyond (k–1) level

Why ⇒ there are only (k –1) smaller elements than k^{th} smallest and path from root to k^{th} smallest must go through these (k–1) elements.

Ensure: Enough nodes

Level 0 → $2^o = 1$

Level 1 → $2^1 = 2$

upto level 1 = $2^{1+1} - 1 = 3$

upto level 2 ⇒$2^{2+1} - = 7$

:

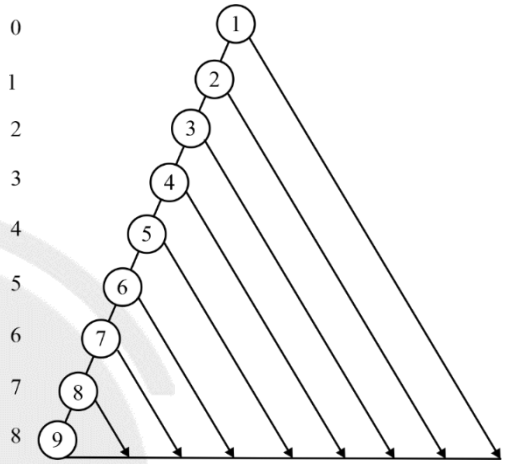
:

upto level 7 ⇒ $2^{7+1} - 1 = 2^8 - 1 = 255$

upto level 8^{th} ⇒ $2^{8+1} = 511$

We are having more than 511.

⇒ Deepest level for 9 is 8



So, 8 is the correct answer.

Scan for Video solution

8. (b)

Delete (i) → delete ith element

O(1) time

But we need to maintain heap property after deletion

To delete i^{th} element ⇒ replace it with last element

i.e. A[i]↔ A [n]

The, n = n –1

Apply Heapify algorithm to re- fix the heap

It takess ⇒ O (logn) time in worst case

But here it is given that d is the depth

So, heapify will take O(d).

Therefore, option (b) is the correct answer.

Scan for Video solution

9. (b)

Given code is Implementation of binary search whenever we found the target element i.e. x both if conditions become true.

Code for first if:

$j = k - 1 \Rightarrow$ update j to $k - 1$

Code of second if:

$i = k + 1$

$$\left.\begin{array}{l} j = k - 1 \\ i = k + 1 \end{array}\right\}$$

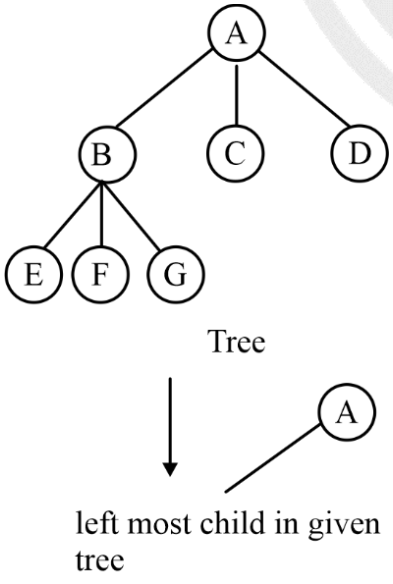
$\Rightarrow i > j$

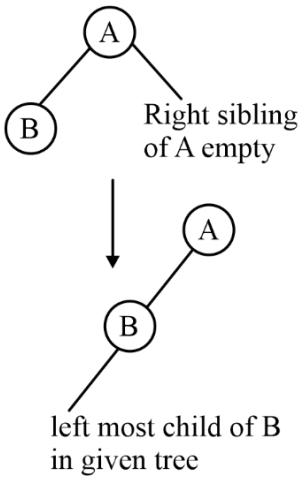
$\Rightarrow$ Condition inside while loop become false and we come out of loop.

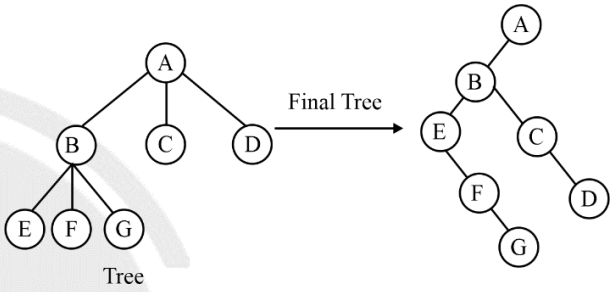
So, given code is an implementation of binary search.

Scan for Video solution

10. (d)







Current node is a leaf node

// if left most child pointer is NULL

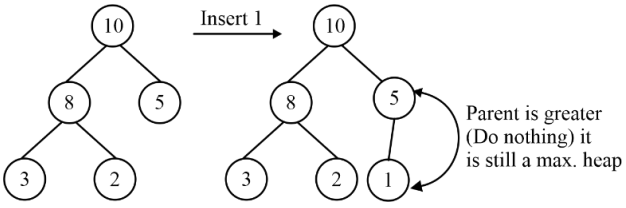
$\Rightarrow$ there is no child of node under consideration.

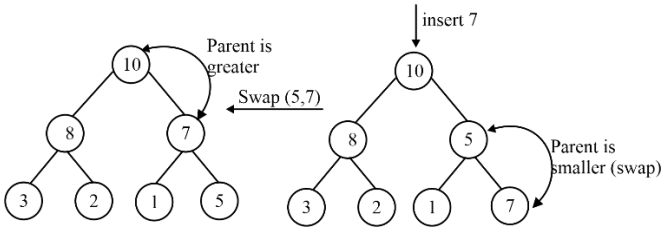
If there is no left child we are counting the number of leaf nodes.

Scan for Video solution

11. (a)

Initial:



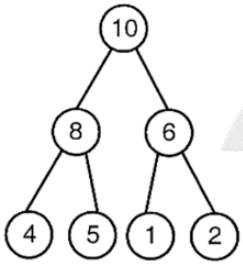


Level order traversal of max heap

= 10, 8, 7, 3, 2, 1, 5

Scan for Video solution

12. (b)



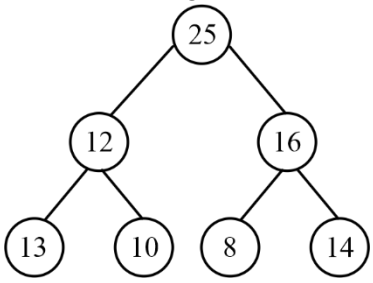
Max– heap

Satisfy max heap property only option (b)

Scan for Video solution

3. (c)

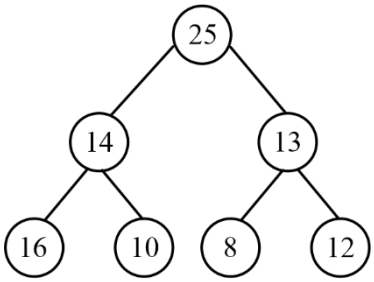
(a) Parent must be greater.



13 is greater than 12 (Parent)

Not a max-heap

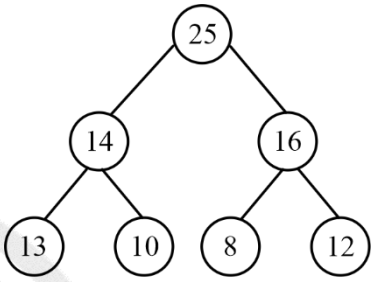
(b)



Not a max– heap

Not a max heap because 16 must be parent of 14 and 10.

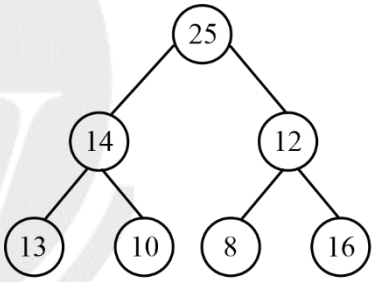
(c)



All nodes satisfy Property

Max- heap

(d)

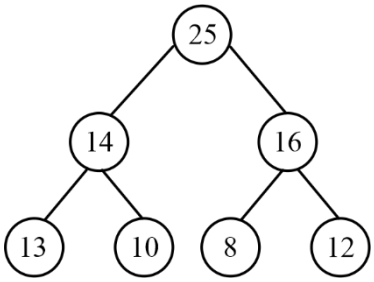


12 is smaller than its right child 16

Not a max- heap

Scan for Video solution

14. (d)



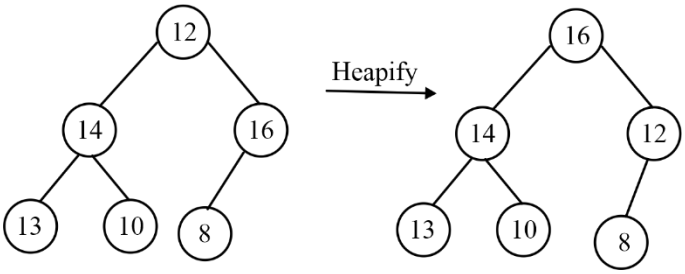
1	2	3	4	5	6	7
25	14	16	13	10	8	12

n = 7

A = [1] ↔ A [n] swap root element with last element.

n = n –1

heapify on A [1]

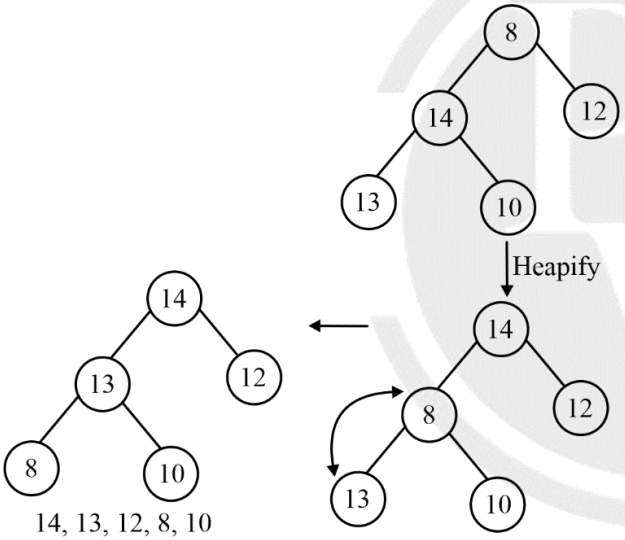


another delete

(i) swap 12, 16

(ii) n = n–1

(iii) Heapify



Scan for Video solution

15. (b)

1 insert ⇒O (logn)

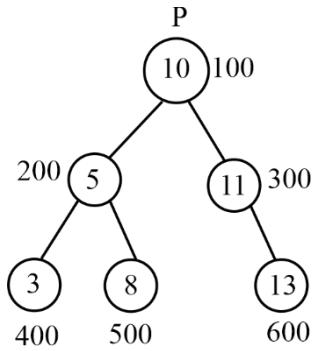
(i) Inset n element without anything

(ii) Apply build heap algorithm on 2n element

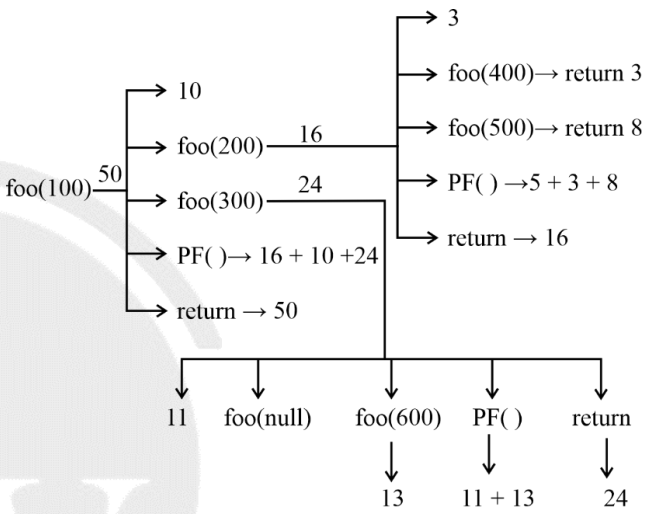
Time Complexity = O(2n) = O(n)

Scan for Video solution

16. (c)



P is a root node



Output = 3, 8, 16, 13, 24, 50

Output matched with option (c)

Scan for Video solution

17. (c)

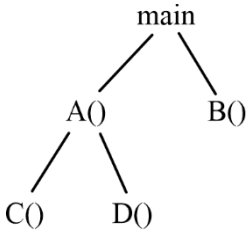
S1 : Ture

To perform func/procedure calls.

A functioni/procedure (parent) must call child functions

```
Main A( )
 {
A( );
B( );
}
A( )
{
C( );
```

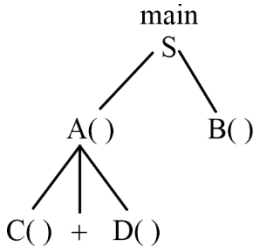
```
D( );
}
C( )
{
-
-
}
D( )
{
-
-
}
```



S_2: True because in order that parent func return its vaule, its child must provide (return) value to it.

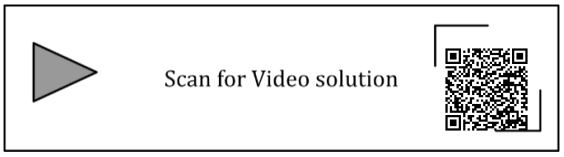
```
Main ()
{
Int s;
S = A( ) + B( )
Pf ("%d, S")
}
Int A( )
{
Return (C) + D( );
}
Int B( )
{
Retrun 1;
}
int
C( );
{
return 2
}
int D( )
{
return 3;
}
```

Procedure return call:



Procedure return call = C ( ), D( ), A( ), B( ), S

It is nothing but post order.



18. (b)

The in-order traversal of a BST is increasing order of keys.

In order:	10	11	12	15	16	18	19	20
Pr order:	15	10	12	11	20	18	16	19

we can construct the BST

We know, first element of preorder traversal is the root node of entire tree.

Mark this node in Inorder traversal

Now we will insert 10 (next node in preorder) in the BST.

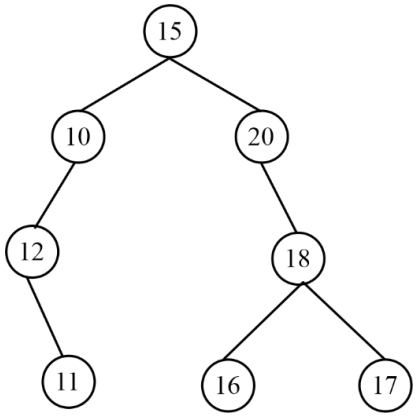
$\Rightarrow$ to find the position of 10

$\Rightarrow$ just check position of 10 in Inorder traversal w.r.t. the marked node.

{10 is the left of 15}

Similarly, finding position for each other element

Now we can find the post-order traversal



Post-order traversal: 11,12, 10, 16, 19, 18, 20, 15.

Therefore, option B is the correct answer.

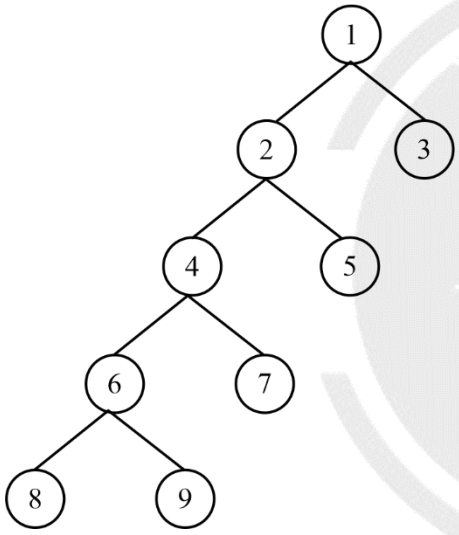
Scan for Video solution

19. (4 to 4)

In order: 8 9 6 7 4 5 2 3 1

Pr order: 8 6 9 4 7 2 5 1 3

We will construct the binary tree using these 2 traversals & then find the height of binary tree.



Height = 4 of the tree.

Scan for Video solution

20. (b)

The Inorder traversal of a BST is increasing order of keys.

Inorder: Left subtree- Root Node- Right subtree

Pre-order: Root Node- Left subtree - Right subtree

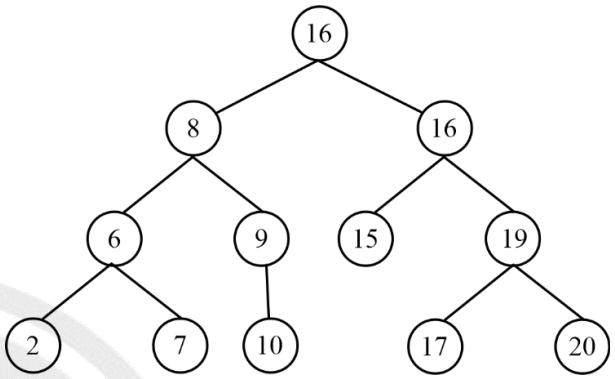
L_T ,Root, R_T 2 6 7 8 9 10 12 15 16

Root, L_T, R_T 12 8 6 2 7 9 10 16 15

17 19 20

19 17 20

insert nodes into initially empty BST in this order



Post-order traversal of this tree: 2, 7, 6, 10, 9, 8, 15, 17, 20 19, 12. So, option (b) is the correct answer.

Scan for Video solution

21. (a)

In-order traversal of a BST is increasing order of keys.

1. Increasing order of keys (✓)
2. Not increasing order of key cannot be in-order traversal of a BST (×)
3. Not increasing order of keys (×)
4. Increasing order of keys (✓)

Only 1 and 4 are Inorder traversals, therefore, option a is correct answer.

Scan for Video solution

22. (d)

BST: The in-order traversal is increasing order of keys.

In order: 10 15 20 23 25 **30** 35 39 42

Pr order: 30 20 10 15 25 23 39 35 42

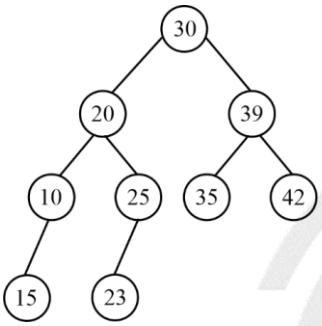
Insert nodes in the tree from left to right in pre-order.

Because pre-order gives first node as a root nods

In order: 10 15 20 23 25 30 35 39 42

Pr order: 30 20 10 15 25 23 39 35 42

Tree:



Post order = 15, 10, 23, 25, 20, 35, 42, 39, 30

Scan for Video solution

23. (d)

I. MBCAFHPYK **Preorder:** Root, L_T, R_T

II. KAMCBYPFH **Postorder:** L_T, R_T, Root

III. MABCKYFPH

Let I be preorder

Root ⇒ M

Out of II and III one must be the postorder & therefore M must be the lost node in any then but it is not.

Hence, I is not preorder

Let II be preorder

⇒ K is root node I has k in last

⇒ I is postorder

& remaining trarversal i.e. III is Inorder.

Scan for Video solution

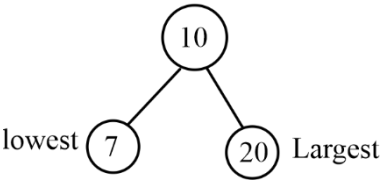
24. (509 to 509)

BST: keys are stored in some orderd way

The in-order traversal is increasing order of keys.

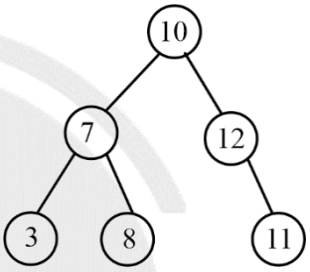
1. Consider an BST

Medium



In - order : 7 10 20

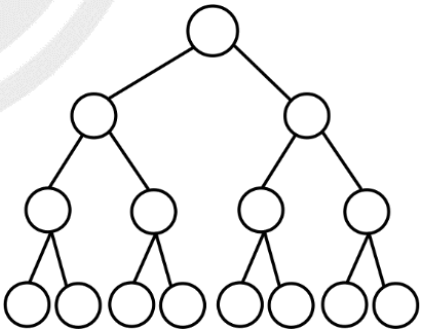
2.Take another BST



In – Order = 3 7 8 11 11 12

For a binary tree of height h, if all levels are full,then number of nodes = $2^{h+1} - 1$

If a height is 3 of CBT then number of nodes must be $2^4 - 1 = 15$



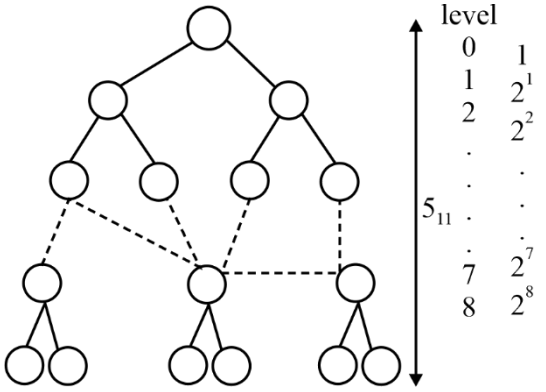
Height = 3

Level = 4

Number of nodes = 15

If in CBT having L level then number of nodes = $2^L - 1$

If N = 1000, all the levels are not full.



Till Height 8 or level 9 Number of nodes = $2^9 - 1$

Number of nodes = 511

For last level to be full number of nodes = $2^9 = 512$

Number of nodes remaining = 1000 – 511 = 489

In second last level (256 Nodes) 244 nodes having 2 child and 1 node having only 1 child and remaining are countion in leaf node.

3 largest element index value must be 509.

Scan for Video solution

25. (d)

We want any element other than maximum or we can say any element which is not maximum would be our element.

So, we need to pick any 2 elements starting from root node, then return the smaller from these 2 element.

⇒ Complexity would be Θ(1)

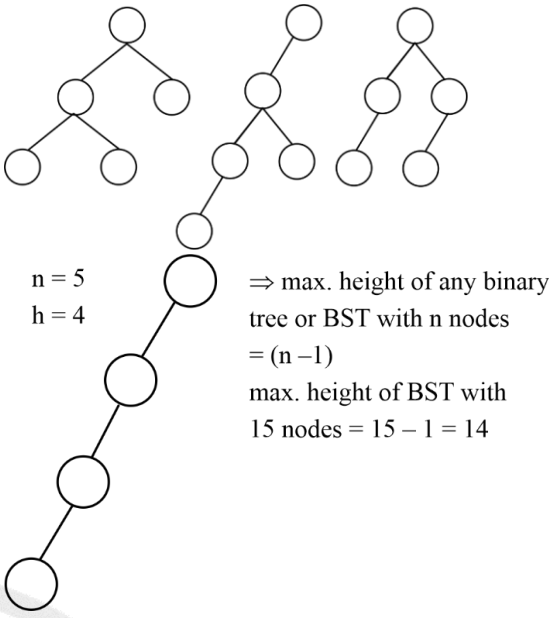
Scan for Video solution

26. (b)

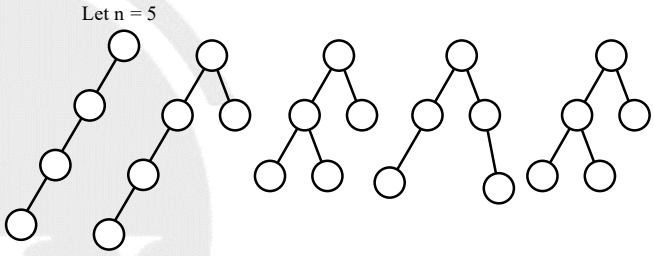
Let n = 5

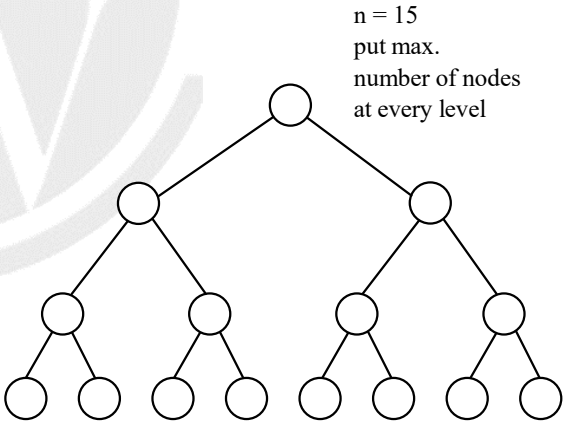
Maximum Height can be achieve when minimum nodes are present at every level.

i.e. only 1 node at each level.



Min. height of a binary tree with 5 nodes





Minimum height =3

Maximum height = 14

Therefore, option B is the correct answer.

Scan for Video solution

27. (64 to 64)

We cannot have any node with 2 child nodes.

At every level ⇒ 1 node can be there (either to left or to right)

From 1st level till 2nd last level⇒ 2 choices are there

Last level ⇒ only one choice

For six levels: 2×2×2×2×2×2×1 = 64

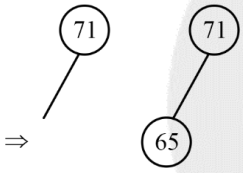
Scan for Video solution

28. (b)

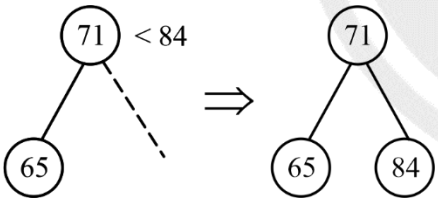
71, 65, 84, 69, 67, 83

(1) Insert 71

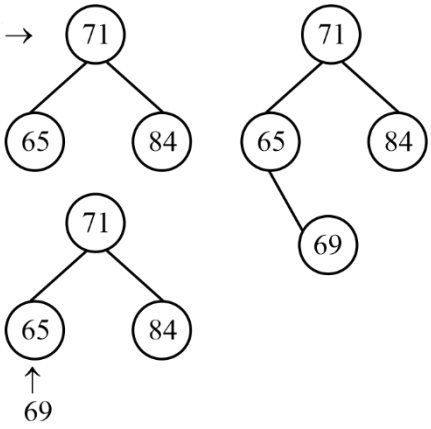
(2) Insert 65 65<



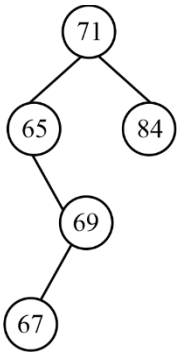
(3) Insert 84



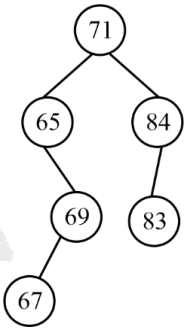
(4) Inset 69



(5) Insert 67



(6) Insert 83



⇒Lowest level element= 67. So, option (b) is the correct answer.

Scan for Video solution

29. (19 to 19)

Number of nodes with two child = Number of nodes with no child (leaf node) –1

$$n_2 = n_0 - 1$$

$n_2 = 20 - 1$

$n_2 = 19$

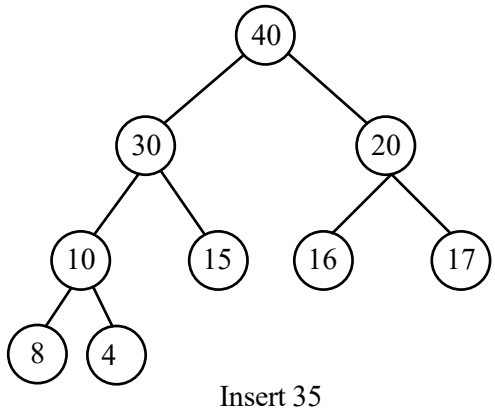
So, 19 is the correct answer

Scan for Video solution

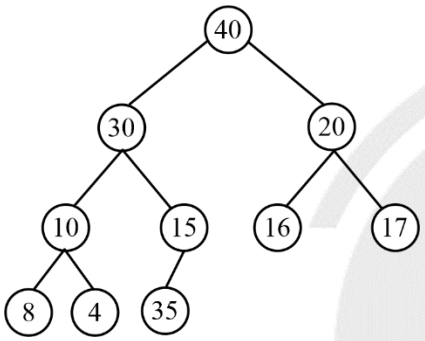
30. (b)

Given, 40, 30, 20, 10, 15, 16, 17, 8, 4

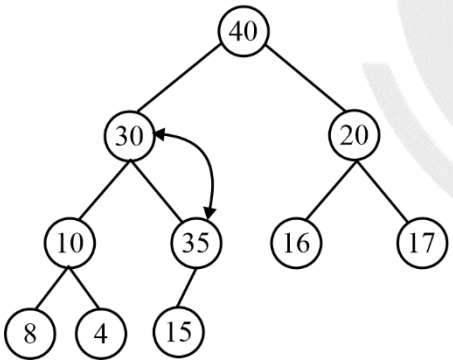
Corresponding heap:



Insert 35

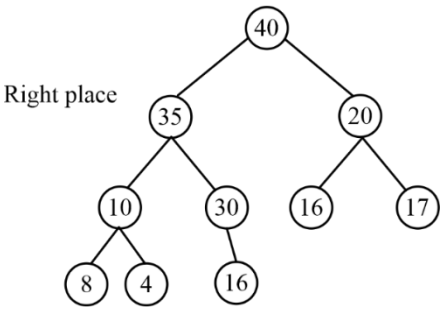


Swap

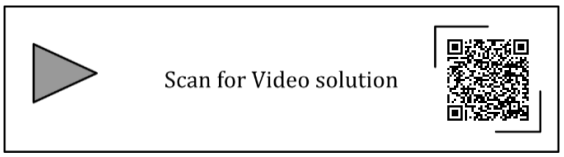


Swap

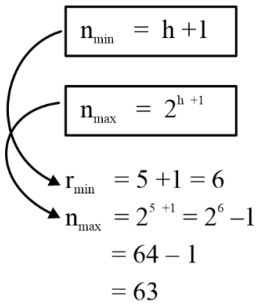
Right place



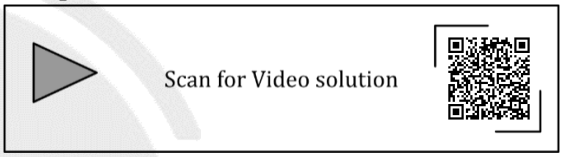
So, 40, 35, 20, 10, 30, 16, 17, 8, 4, 15. Option b is the correct answer.



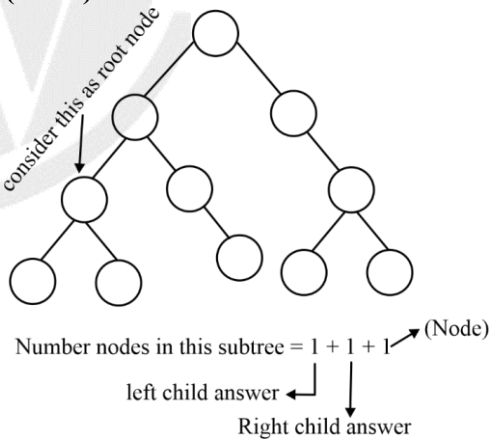
31. (a)



So, option a is the correct answer.



32. (1 to 1)



we need to use bottom up approach

```
Count = 0; // global
int f (struct node * Root)
{
if (Root = = NULL)
return 0 ;
int l = f (Root → left);
```

```
int r = f (Root → Right);
if (l + r +1 = = 4)
count + +;
return l + r + 1;
}
```

Time = O(n)

Given time = $O\left(n^a \log^b n\right)$

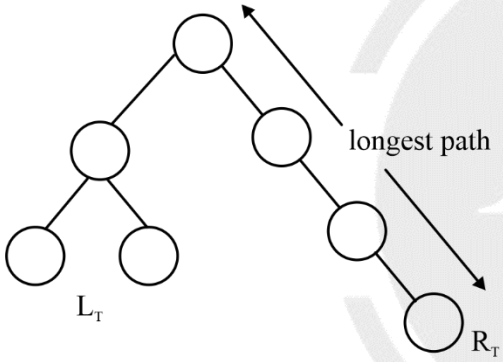
a = 1

b = 0

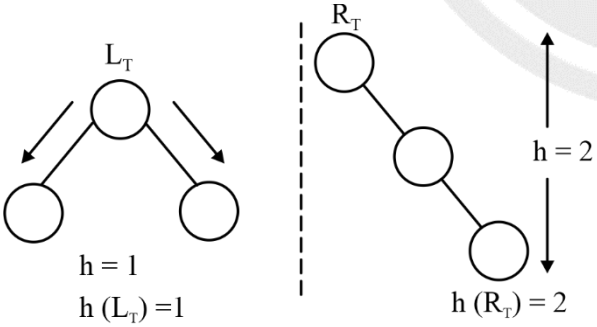
$O\left(n^a \log^b n\right) = a + 10 \times b = 1 + 10 \times 0 = 1$

Scan for Video solution

33. (a)



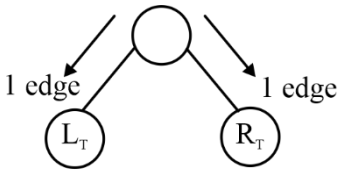
Height (h) = 3



h = 1

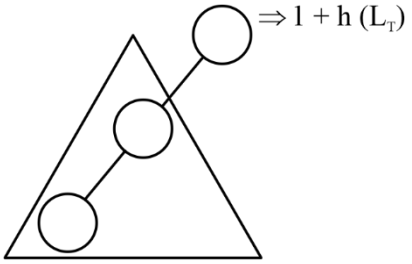
$h(L_T) = 1$

$h(R_T) = 2$



$h = 1 + \max(h(L_T), h(R_T))$

- If L_T = empty and R_T = empty



$h(L_T) = 1$

If R_T = = NULL

$h = 1 + h(L_T)$

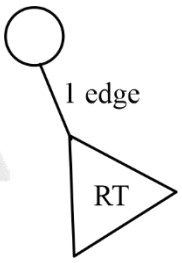
If L_T = = NULL

$h = 1 + h(R_T)$

- $R_T, L_T \Rightarrow$ non - empty

$1 + \max(h(L_T), h(R_T))$

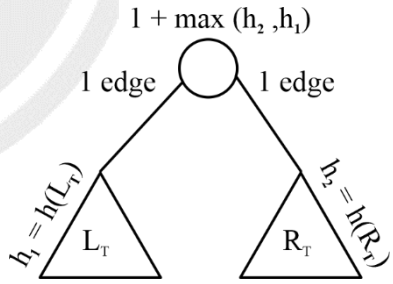
B1:



$h = 1 + h(R_T)$

B1: 1 + height (n → Right)

B2:



$B2 = 1 + \max(h_1, h_2)$

Scan for Video solution

34. (b)

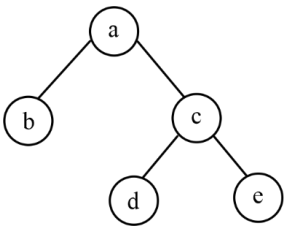
- Given a binary tree structure with n nodes and also given n distinct elements.
- Let $n = 5$

10, 13, 18, 19, 35

Because it is BST,

keys are ordered in some way.

Smallest element must be to the left of root node (1 choice)



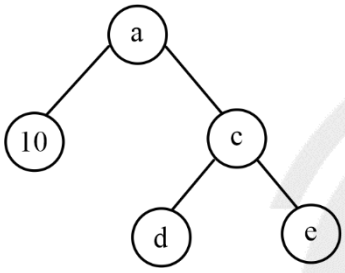
b < a < (c, d, e)

$\Rightarrow \boxed{b=10}$

13, 18, 19, 35 are remaining

a < all keys in right subtree

among remaining keys, the smallest is the root node

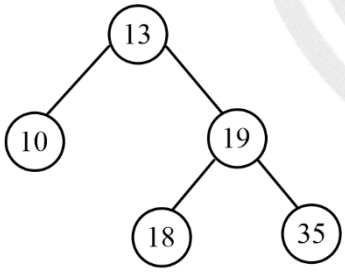


$\boxed{a=13}$

Remaining: 18, 19, 35

d < c < e

$18<19<35$ {only1 choice}



$\Rightarrow$ only 1 way

Hence, option (b) is correct.

Scan for Video solution

35. (b)

BST: The in-order traversal is increasing order of keys.

In-order: 1, 2, 3, 4, ……..n

Post-order ----------Root

(i) from Post-order $\rightarrow$ lost node is the root node O(1) time.

Finding this root node in in-order traversal (sorted array)

BS$\rightarrow$ O($\log_n$)

In-order: 1, 2, 3, 4……n

Post order: _____Root

(1) From Post-order $\rightarrow$last node is the root node O(1)

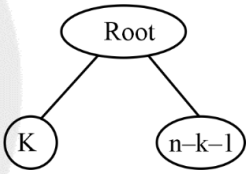
Finding this root node in inorder traversal (if sorted array then, BS$\rightarrow$ O($\log_n$))

$\Rightarrow$ Split the postorder
- Smaller than pivot (left)
- Greater than pivot (Right)

If k element in left subtree then in right subtree element must be n – k – 1.

(n–k –1)

$T(n) = O(\log_n) + T(k) + T(n-k-1)$



Recursively follow.

$T(n) = T(k) + T(n-k-1) + O(\log_n)$

$T(n) = O(n \log n)$

But

In-order [ ]

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

$\underbrace{1,2,3,....n}_{\text{let}}$ root

we can find any element in O(1) time

Left (1…. 5) 6 (7….10)

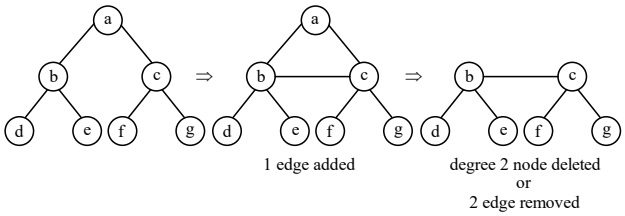
T(n) = O(1) + T(k) + T(n – k –1)

Hence, option (b) is correct.

Scan for Video solution

36. (a)

$\boxed{n_3 = n_1 - 2}$ Already know



In 1 such removal ⇒ 1 edge remove

Repeat this process until all nodes of degree 2 removed

⇒ n_2 times we need to repeat

⇒ $n_2 \times 1$ edge will be removed in over all process

Number of edges initially = $n - 1 = n_1 + n_2 + n_3 - 1$

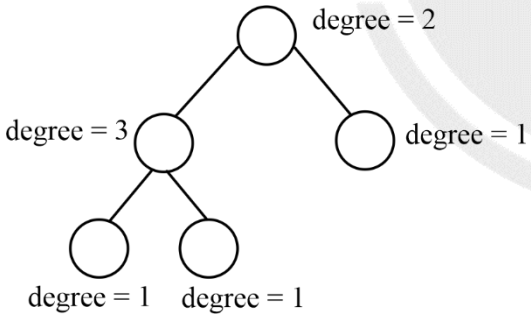
After complete process edges, remaining = $(n_1 + n_2 + n_3 - 1) - 1 \times n_2$

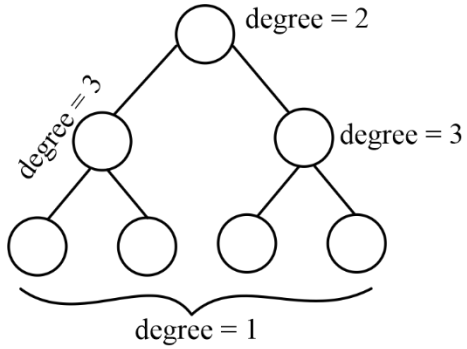
$= n_1 + n_3 - 1 = n_1 + n_1 - 2 - 1 = 2 * n_1 - 3$

Scan for Video solution

37. (b)

Basics:





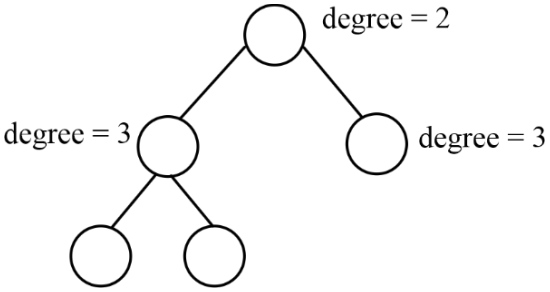
$n_1 = 4$ $n_2 = 2$ $n_3 = 2$

$4 + 1 - 1 = 4$ (False) n_3 most be 2

(b) $4 - 2 = 2$ (True)

(c) [5/2] = [2.5] = Either 2 or 3

(d) 0



$n_1 = 3$, $n_2 = 1$, $n_3 = 1$,

(a) 1

(b) $(3 + 1)/2 = 2$ cannot be our answer.

Method 2:

Total number of nodes $n = n_1 + n_2 + n_3$

Sum of degree of all the nodes = $2|E|$

$= 2(n - 1)$

$= 2(n_1 + n_2 + n_3 - 1)$

$= n_1 \times 1 + n_2 \times 2 + n_3 \times 3$

$n_1 + 2n_2 + 3n_3 = 2n_1 + 2n_2 + 2n_3 - 2$

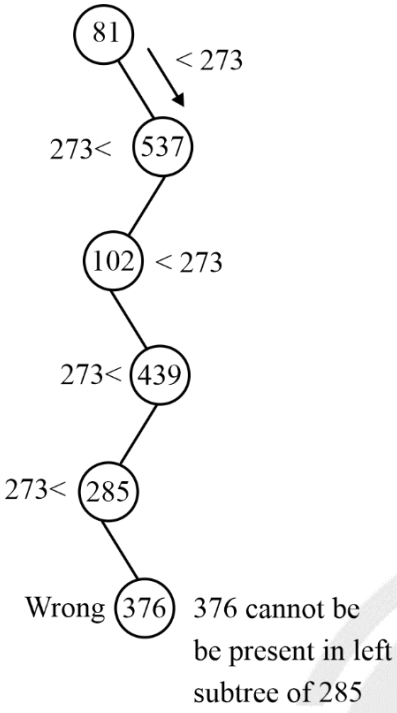
$n_3 = 2n_1 - n_1 - 2$

$n_3 = n_1 - 2$

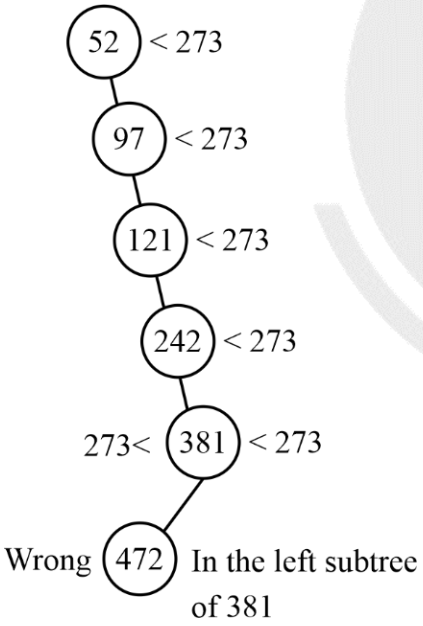
So, option (b) is correct.

Scan for Video solution

38. (d) I.

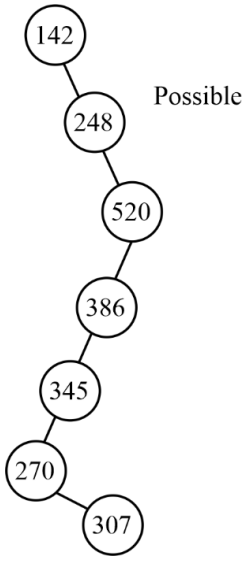


II.

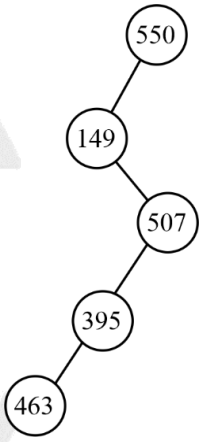


All keys must be smaller than 381.

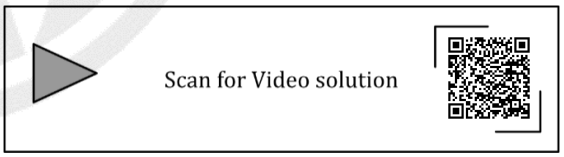
III. 142, 248, 520, 386, 345, 270, 307



IV. 550, 149, 507, 395, 463, 402, 270



463 cannot be in left subtree of 395.



39. (c)

AVL tree is height balanced tree and the height of an AVL tree with n element is $O(\log_2 n)$

1st insertion time = O(logn)

2nd insertion time = O(log(n + 1))

As AVL tree contain (n + 1) elements after 1st insertion

3rd insertion = O(log(n + 2))

.

.

n^2 insert = $O(\log (n + n^2 - 1))$

Total T.C = O(logn) + O(log (n + 1) + O(log (n + 2)) +……+ O(log(n + n^2 – 1))

= Olog (n * (n + 1)) * (n + 2) * … * (n + n^2 – 1)

$O\left(\log n^{n^2}\right)$

$O(n^2 \log n)$

(c) is correct

Scan for Video solution

40. (b)

n(h) = 1+ n (h–1) + n(h –2)

minimum nodes with height h.

n (0) =1

n (1) = 2

n (2) = 1+ n(1) +n(0) = 1+2 +1 = 4

n(3) = 1 + n(2) + n(1)

= 1 + 4 + 2 = 7

$\boxed{n(3)=7}$

Scan for Video solution

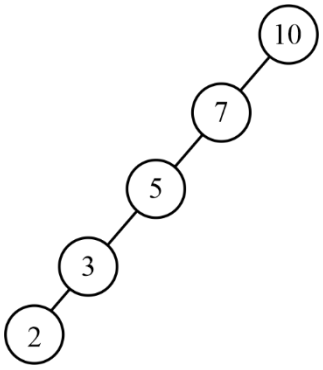
41. (a)

AVL tree is height balance tree (also a search tree)

Height of AVL tree = $O(\log_2 n)$

searching time = $O(\log_2 n)$

Binary search tree:



Searching time = O(n)

The height of complete binary tree = $O(\log_2 n)$ but it is not a search tree. Element order arranged randomly (not in some ordered way)

Searching time = O(n)

Scan for Video solution

❑❑❑

CHAPTER

8

HASHING

Hash Functions

1. [MCQ] [GATE-2022 : 1M]

Suppose we are given n keys, m hash table slots, and two simple uniform hash functions h_1 and h_2 . Further suppose our hashing scheme uses h_1 for the odd keys and h_2 for the even keys. What is the expected number of keys in a slot?

(a) $\frac{m}{n}$ (b) $\frac{n}{m}$

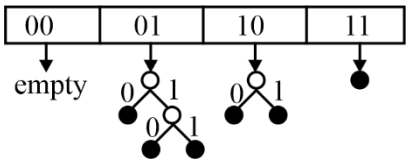
(c) $\frac{2n}{m}$ (d) $\frac{n}{2m}$

2. [MSQ] [GATE-2021 : 2M]

Consider a dynamic hashing approach for 4-bit integer keys:

1. There is a main hash table of size 4
2. The 2 least significant bits of a key is used to index into the main hash table.
3. Initially, the main hash table entries are empty.
4. Thereafter, when more keys are hashed into it, to resolve collisions, the set of all keys corresponding to a main hash table entry is organized as a binary tree that grows on demand.
5. First, the 3rd least significant bit is used to divide the keys into left and right subtrees.
6. To resolve more collisions, each node of the binary tree is further sub-divided into left and right subtrees based on the 4th least significant bit.
7. A split is done only if it is needed, i.e., only when there is a collision.

Consider the following state of the hash table.



Which of the following sequences of key insertions can cause the above state of the hash table (assume the keys are in decimal notation)?

(a) 5, 9, 4, 13, 10, 7

(b) 9, 5, 10, 6, 7, 1

(c) 10, 9, 6, 7, 5, 13

(d) 9, 5, 13, 6, 10, 14

3. [NAT] [GATE-2020 : 1M]

Consider a double hashing scheme in which the primary hash function is $h_1(k) = k \bmod 23$ and the secondary hash function is $h_2(k) = 1 + (k \bmod 19)$. Assume that the table size is 23. Then the address returned by probe 1 in the probe sequence (assume that the probe sequence begins at probe 0) for key value $k = 90$ is _____

4. [NAT] [GATE-2015 : 1M]

Given a hash table T with 25 slots that stores 2000 elements, the load factor α for T is _____.

5. [MCQ] [GATE-2015 : 2M]

Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?

(a) h(i) = i^2 mod 10

(b) h(i) = i^3 mod 10

(c) h(i) = (11 * i^2) mod 10

(d) h(i) = (12 * i) mod 10

Collision Resolution Techniques

6. [MCQ] [GATE-2023 : 1M]

An algorithm has to store several keys generated by an adversary in a hash table. The adversary is malicious who tries to maximize the number of collisions.

Let k be the number of keys, m be the number of slots in the hash table, and k > m. Which one of the following is the best hashing strategy to counteract the adversary?

(a) Division method, i.e., use the hash function h(k) = k mod m.

(b) Multiplication method, i.e., use the hash function $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$, where A is a carefully chosen constant.

(c) Universal hashing method.

(d) If k is a prime number, use Division method. Otherwise, use Multiplication method.

7. [MCQ] [GATE-2014 : 2M]

Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

(a) $(97 \times 97 \times 97)/100^3$

(b) $(99 \times 98 \times 97)/100^3$

(c) $(97 \times 96 \times 95)/100^3$

(d) $(97 \times 96 \times 95)/(3! \times 100^3)$

8. [MCQ] [GATE-2014 : 2M]

Consider a hash table with 9 slots. The hash function is h(k) = k mod 9. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are

(a) 3, 0, and 1 (b) 3, 3 and 3

(c) 4, 0 and 1 (d) 3, 0, and 2

Common Data for next two questions (9-10):

A hash table of length 10 uses open addressing with hash function h(k) = k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

9. [MCQ] [GATE-2010 : 2M]

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(a) 46, 42, 34, 52, 23, 33

(b) 34, 42, 23, 52, 33, 46

(c) 46, 34, 42, 23, 52, 33

(d) 42, 46, 33, 23, 34, 52

10. [MCQ] [GATE-2010 : 2M]

How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table shown above?

(a) 10 (b) 20

(c) 30 (d) 40

11. [MCQ] [GATE-2009 : 2M]

The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table?

(a)

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(b)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(c)

0	
1	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(d)

0	
1	
2	12,2
3	13,3,23
4	
5	5,15
6	
7	
8	18
9	

12. [MCQ] [GATE-2008 : 2M]

Consider a hash table of size 11 that uses open addressing with linear probing. Let h(k) = k mod 11 be the hash function used. A sequence of records with keys 43 36 92 87 11 4 71 13 14 is inserted into an initially empty hash table, the bins of which are indexed from zero to ten. What is the index of the bin into which the last record is inserted?

(a) 2 (b) 4

(c) 6 (d) 7

❑❑❑

ANSWER KEY

1.	(b)	**2.**	(c)	**3.**	(13 to 13)	**4.**	(80 to 80)
5.	(b)	**6.**	(c)	**7.**	(a)	**8.**	(a)
9.	(c)	**10.**	(c)	**11.**	(c)	**12.**	(d)

SOLUTIONS

1. (b)

Either h_1 or h_2 for any key k every slot is equal likely.

Each bucket /slot is equal likely occupied.

For every slot the probability that key will be occupied = $\frac{1}{m}$

Expected number of keys in the slots = $n \times \frac{1}{m} = \frac{n}{m}$

Scan for Video solution

2. (c)

Given

00 = empty

01 = 3 keys

10 = 2 keys.

11 = 1key

(a) 5 = 0101 01 = keys (5, 9, 15)

9 = 1001 00 = 1key fail it must be empty.

4 = 0100 10 = 1 key (must be 2 key)

13 = 1101 11 = 1 key

10 = 1010

7 = 0111

(b) 9 = 1001 01 = 3 keys

5 = 0101 10 = 2 keys

10 = 1010 11 = 1 key

6 = 0110

7 = 0111

1 = 0001

According to 3rd bit in 01, 1 entries in right and 2 in left but given hash table having 2 in right and 1 in left.

(c) 10 = 1010 01 = 3 keys

9 = 1001 10 = 2 keys

6 = 0110 11 = 1 key

7 = 0111

5 = 0101

13 = 1101

According to 3rd bit in 01 entry 2 key in right and 1 in left.

If is matched with given hash table.

Scan for Video solution

3. (13 to 13)

m = 23

$h_1(k)$ = k mod 23 = Primary hashing

$h_2(k)$ = 1 + (k mod 19) = Secondary hashing

H(k, i) = $(h_1(k) + i.h_2(k)) \bmod m$

i = Collision number

k = 90

$h_1(k)$ = 90 mod 23 = 21

$h_2(k) = 1 + (90 \bmod 19)$

$= 1 + 14 = 15$

$H(k, i) = (h_1(k) + i.h_2(k)) \bmod m$

$H(90, i) = (h_1(90) + i.h_2(90) \bmod 23$

$= (21 + 15)$ mode 23

$= 13$

13 is the right answer

Scan for Video solution

4. (80 to 80)

Formula: $\alpha = \dfrac{\text{Number of keys (n)}}{\text{Table size (m)}}$

Number of keys = 2000

Table size (m) = 25

$\alpha = \dfrac{2000}{25} \Rightarrow$

$\boxed{\alpha = 80}$

Scan for Video solution

5. (b)

Number of buckets = 10

m = 10 [0 to 9]

(a) $1^2 = 1 \bmod 10 = 1$

$2^2 = 4 \bmod 10 = 4$

$3^3 = 9 \bmod 10 = 9$

$4^2 = 16 \bmod 10 = 6$

$5^2 = 25 \bmod 10 = 5$

$6^2 = 36 \bmod 10 = 6$

$7^2 = 49 \bmod 10 = 9$

$8^2 = 64 \bmod 10 = 4$

$9^2 = 81 \bmod 10 = 1$

$10^2 = 100 \bmod 10 = 0$

Bucket that is filled = 1,4,9,6,5,0

Some buckets remain empty. So, option (a) is wrong as well as option (c) also wrong.

(b) $1^3 = 1 \bmod 10 = 1$

$2^3 = 8 \bmod 10 = 8$

$3^3 = 27 \bmod 10 = 7$

$4^3 = 64 \bmod 10 = 4$

$5^3 = 125 \bmod 10 = 5$

$6^3 = 216 \bmod 10 = 6$

$7^3 = 343 \bmod 10 = 3$

$8^3 = 512 \bmod 10 = 2$

$9^3 = 729 \bmod 10 = 9$

$10^3 = 1000 \bmod 10 = 0$

All buckets will be filled uniformly.

(d) In option (d) all odd number of buckets will be empty

Scan for Video solution

6. (c)

Division method:-

- $h(k) = k \bmod m$

  $m = 2^P$

  $m = 2^4$

  $h(100) = 100 \bmod 2^4 = 100 \bmod 16 = 4$

  $h(128 + 4)$

  $= h(132) = 132 \bmod 16 = 4$

  $132 = 1011100$

  Last 4 bits is deciding the slot.

  A is not correct

- $h(k) = Lm(kA \cdot kA)$

  $m = 100$

  $A = \dfrac{1}{3}$

  $h(10) = \lfloor m(KA.\lfloor KA \rfloor) \rfloor$

  $h(11) = 66$

  $h(12) = 99$

B is wrong we will get only 3 values of h(k)

Optimal choice of a depends of keys themseleve

$$A = \frac{\sqrt{5}-1}{2}$$

- Malicious adversary can always choose keys. So, that all the keys are mapped to same slot ⇒ O(n).

Worst case retrieval time Set of uniform hash function Minimum collision Randomly hash function are picked.

Scan for Video solution

7. (a)

Number of slots = 100

m = 100

- First 3 slots must be empty.
- Number of insertions = 3

$$\text{Probability} = \frac{97}{100} \times \frac{97}{100} \times \frac{97}{100}$$

$$= \frac{97 \times 97 \times 97}{100 \times 100 \times 100}$$

Matched with option (a)

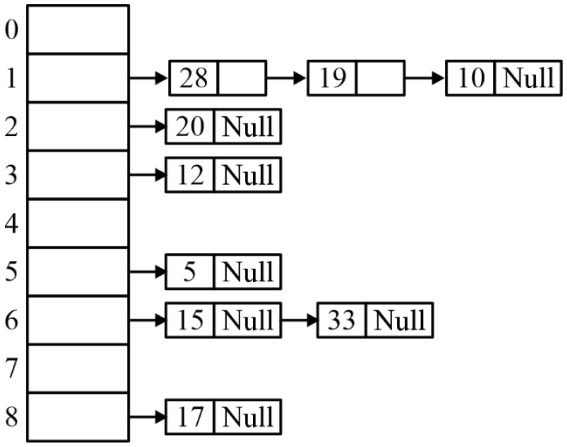
Scan for Video solution

8. (a)

Number of slots = 9 [0 to 8]

m = 9

h (k) = k mod 9



h (5) = 5 mod 9 = 5

h (28) = 28 mod 9 = 1

h (19) = 19 mod 9 = 1

h (15) = 15 mod 9 = 6

h (20) = 20 mod 9 = 2

h (33) = 33 mod 9 = 6

h (12) = 12 mod 9 = 3

h (17) = 17 mod 9 = 8

h (10) = 5 mod 9 = 1

Maximum keys in a slot = 3

Minimum keys in a slot = 0

Average keys in a slot = $\frac{9}{9} = 1$

(3,0 and 1) is a right answer.

Scan for Video solution

9. (c)

Solve with the option

(a)

46 mod 10 = 6
42 mod 10 = 2
34 mod 10 = 4
52 mod 10 = ~~2~~ = 3

0	
1	
2	42
3	52
4	34
5	
6	46
7	
8	
9	

Not matched with given Hash Table.

(b)

34 mod 10 = 4
42 mod 10 = 2
23 mod 10 = 3
52 mod 10 = ~~2 3 4~~ = 5
38 mod 10 = ~~3 4 5~~ = 6

0	
1	
2	42
3	23
4	34
5	52
6	33
7	
8	
9	

(c)

46 mod 10 = 6
34 mod 10 = 4
42 mod 10 = 2
23 mod 10 = 3
52 mod 10 = ~~2 3 4~~ = 5
33 mod 10 = ~~3 4 5 6~~ = 7

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Matched with given Hash Table

Scan for Video solution

10. (c)

- No collision for keys 46, 34, 42 and 23 to enter 52 in hash table total 6 possibilities. After insert 52 in table, to enter key 23 there are 5 possibilities.

Total number of possibilities = 6 × 5 = 30

Scan for Video solution

11. (c)

h (k) = k mod 10
h (12) = 12 mod 10
= 2
h (18) = 18 mod 10
= 8
h (13) = 13 mod 10
= 3
h (2) = 2 mod 10
= 2 ~~3~~ = 4
h (3) = 3 mod 10
= ~~3 4~~ = 5
h (23) = 23 mod 10
= ~~3 4 5~~ = 6
h (5) = 5 mod 10
= ~~5 6~~ = 7
h (15) = 15 mod 10
= ~~5 6 7 8~~ = 9

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

Match with option (c)

Scan for Video solution

12. (c)

h (k) = k mod 11
h (43) = 43 mod 11
= 10
h (36) = 36 mod 11
= 3
h (92) = 92 mod 11
= 4
h (87) = 87 mod 11
= ~~10~~ = 0
h (11) = 11 mod 11
= ~~0~~ = 1
h (4) = 4 mod 11
= ~~4~~ = 5
h (71) = 71 mod 11
= ~~5~~ = 6
h (13) = 13 mod 11
= 2
h (14) = 14 mod 11
= ~~3 4 5 6~~ = 7

0	87
1	11
2	13
3	36
4	92
5	4
6	71
7	
8	
9	43

Last record position = 7
Correct answer = (d)

Scan for Video solution

❑❑❑