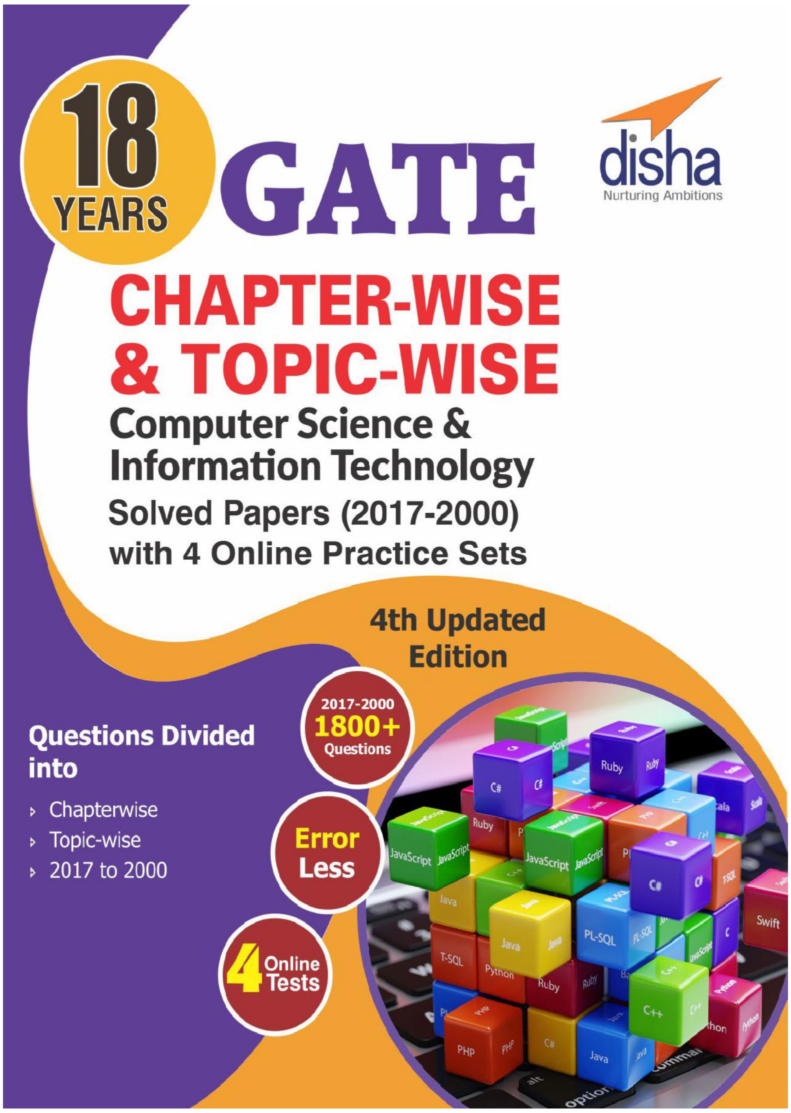
Programming & Data Structures

This Chapter “Programming & Data Structures” is taken from our Book:



ISBN : 9789386629043

PROGRAMMING & DATA STRUCTURE

3

Programming in C

1. Consider the *C* code fragment given below. **[2017, Set 1, 1 Mark]**

```
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.

2. Consider the following intermediate program in three address code **[2017, Set 1, 1 Mark]**

P = a – b
q = P * c
P = u * v
q = p + q

Which one of the following corresponds to a *static single assignment* form of the above code?

(a) $p_1 = a - b$
$q_1 = p_1 * c$
$p_1 = u * v$
$q_1 = p_1 + q_1$

(b) $p_3 = a - b$
$q_4 = p_3 * c$
$p_4 = u * v$
$q_5 = p_4 + q_4$

(c) $p_1 = a - b$
$q_1 = p_2 * c$
$p_3 = u * v$
$q_2 = p_4 + q_3$

(d) $p_1 = a - b$
$q_1 = p * c$
$p_2 = u * v$
$q_2 = P + q$

3. Consider the following C code: **[2017, Set 1, 1 Mark]**

```
#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

4. Consider the following two functions. **[2017, Set 1, 2 Marks]**

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

The output printed when funl (5) is called is

(a) 53423122233445 (b) 53423120112233
(c) 53423122132435 (d) 53423120213243

5. Consider the C functions foo and bar given below: **[2017, Set 1, 2 Marks]**

```
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–l);
  }
  return val;
}
```

Invocations 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.

6. Consider the following C program. **[2017, Set 1, 2 Marks]**

```
#include <stdio.h>
#include <string.h>
void print length (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 ________.

7. The output of executing the following C program is _______. **[2017, Set 1, 2 Marks]**

```
#include <stdio.h>
int total (int v) {
        static int. count = 0;
        while (v) {
           count + = v&l ;
           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) ;
}
```

8. Match the following: **[2017, Set 2, 1 Mark]**

(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 var 1;	(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)

9. Consider the following function implemented in C: **[2017, Set 2, 1 Mark]**

```
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

10. 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. **[2017, Set 2, 2 Marks]**

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

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)

11. Consider the following C function. **[2017, Set 2, 2 Marks]**

```
int. fun (int n) {
        int i,  j ;
        for (i = 1; i <= n; i++)  {
            for (j = 1; j < n; j += i) {
                        printf{“ %d %d”, i, j};
            }
        }
}
```

Time complexity of fun in terms of Θ notation is

(a) $\Theta(n\sqrt{n})$ (b) $\Theta(n^2)$
(c) $\Theta(n \log n)$ (d) $\Theta(n^2 \log n)$

12. Consider the following C Program. **[2017, Set 2, 2 Marks]**

```
#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 _________.

13. Consider the following C Program. **[2017, Set 2, 2 Marks]**

```
#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 ________.

14. Consider the following C program: **[2016, Set 1, 1 Mark]**

```
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)

15. Consider the following C program: **[2016, Set 1, 1 Mark]**

```
#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_______.

16. The following function computes the maximum value contained in an integer array p[ ] of size n (n >= 1). **[2016, Set 1, 2 Marks]**

```
        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

17. What will be the output of the following C program? **[2016, Set 1, 2 Marks]**

```
        void count (int n){
        static int d=1;

        print f (“%d”, n);
        print f (“%d”, d);
        d++;
        if (n>1) count (n – 1);
        print f (“%d”, d);
}
void main( ){
        count(3);
}
```

(a) 3 1 2 2 1 3 4 4 4 (b) 3 1 2 1 1 1 2 2 2

(c) 3 1 2 2 1 3 4 (d) 3 1 2 1 1 1 2

18. What will be the output of the following pseudo-code when parameters are passed by reference and dynamic scoping is assumed? **[2016, Set 1, 2 Marks]**

```
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

19. The attributes of three arithmetic operators in some programming language are given below.

Operator	**Precedence**	**Associativity**	**Parity**
+	High	Left	Binary
–	Medium	Right	Binary
*	Low	Left	Binary

The value of the expression **2 – 5 + 1 – 7 * 3** in this language is _________ . **[2016, Set 1, 2 Marks]**

20. The value printed by the following program is _______. **[2016, Set 2, 1 Mark]**

```
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);
}
```

21. The following function computes X^Y for positive integers X and Y. **[2016, Set 2, 2 Marks]**

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

22. Consider the following program: **[2016, Set 2, 2 Marks]**

```
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 ________ .

23. The output of the following C program is ______. **[2015, Set 1, 1 Mark]**

```
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);
}
```

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

```
begin
q: = 0
r : = x
while r ≥ y do
        being
            r : = r – y
            q : = q + 1
        end
end
```

The post condition that needs to be satisfied after the program terminates is **[2015, Set 1, 2 Marks]**

(a) $\{r = qx + y \wedge r < y\}$

(b) $\{x = qy + r \wedge r < y\}$

(c) $\{y = qx + r \wedge 0 < r < y\}$

(d) $\{q + 1 < r - y \wedge y > 0\}$

25. What is the output of the following C code? Assume that the address of x is 2000 (in decimal) and an integer requires four bytes of memory?

```
int main( )
{
unsigened int x[4] [3] = {(1,2,3), {4,5,6}, {7, 8, 9}, {10, 11,
12}};
printf("%u, %u, %u", x+3, *(x+3),
        *(x+2)+3);
}
```

[2015, Set 1, 2 Marks]

(a) 2036, 2036, 2036

(b) 2012, 4, 2204

(c) 2036, 10, 10

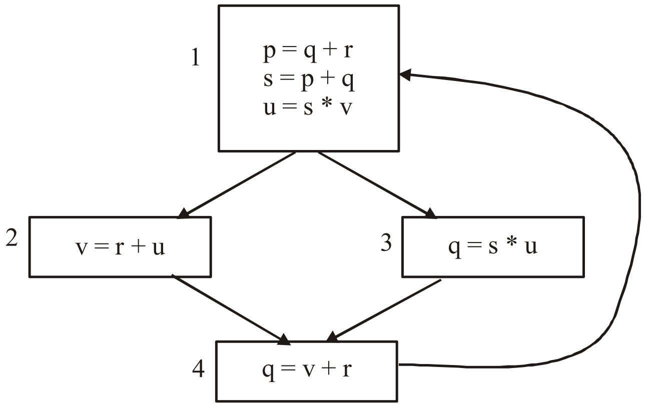
(d) 2012, 4, 6

26. A variable x is said to be live at a statement S_i in a program if the following three conditions hold simultaneously:

I. There exists a statement S_j that uses x

II. There is a path from S_i to S_j in the flow graph corresponding to the program

III. The path has no intervening assignment to x including at S_i and S_j



The variables which are live at the statement in basic block 2 and at the statement in basic block 3 of the above control flow graph are **[2015, Set 1, 2 Marks]**

(a) p, s, u (b) r, s, u

(c) r, u (d) q, v

27. 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 _____. **[2015, Set 2, 1 Mark]**

28. Consider the following function written in the C programming languages.

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

The output of the above function on input "ABCD EFGH" is **[2015, Set 2, 1 Mark]**

(a) ABCD EFGH (b) ABCD

(c) HGFE DCBA (d) DCBA

29. 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)+stkFun(1,0);
}
```

The value printed by the above program is ______.

[2015, Set 2, 2 Marks]

30. Suppose you are provided with the following function declaration in the C programming language.

int partition(int a[ ], int n);

The function treats the first element of a[ ] as s pivot, and rearranges the array so that all elements less than or equal to the pivot is in the left part of the array, and all elements greater than the pivot is in the right part. In addition, it moves the pivot so that the pivot is the last element of the left part. The return value is the number of elements in the left part.

The following partially given function in the C programming language is used to find the kth smallest element in an array a[ ] of size n using the partition function. We assume k ≤ n.

```
int kth_smallest(int a[ ], int n, int k)
{
  int left_end = partition(a, n);
  if (left_end+1 = =k)
{
   Return a[left_end];
}
if(left_end+1) > k)
{
   return kth_smallest(________);
}
else
{
return kth_smallest(________);
}
}
```

The missing argument lists are respectively

[2015, Set 2, 2 Marks]

(a) (a, left_end, k) and (a+left_end+1, n–left_end–1, k–left_end–1)

(b) (a, left_end, k) and (a, n–left_end–1, k–left_end–1)

(c) (a+left_end+1,n–left_end–1, k–left_end–1) and (a, left_end, k)

(d) (a, n–left_end–1, k–left_end–1) and (a, left_end, k)

31. Consider the following C program segment.

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

What will be printed by the program?

[2015, Set 3, 1 Mark]

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

32. 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 time printf statement is executed is ______.

[2015, Set 3, 2 Marks]

33. Consider the following C program.

```
#include<stdio.h>
int f1(void);
int f2(void);
int f3(void);
int x = 10;
int main ( )
    {
    int x = 1;
    x + = f1 ( ) + f2 ( ) + f3( ) + f2( );
    pirntf (“%d”, x);
    retirm 0;
    }
int f1 ( ) { int x = 25; x++; return x;}
int f2 ( ) { static int x = 50; x++; return x;}
int f3 ( ) {x *= 10; return x};
```

The output of the program is ____. **[2015, Set 3, 2 Marks]**

34. Consider the following recursive C function.

```
void get (int n)
{
        if (n<1) return;
        get (n–1);
        get (n–3);
        printf("%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 ( )? **[2015, Set 3, 2 Marks]**

(a) 15 (b) 25

(c) 35 (d) 45

35. 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** ? **[2014, Set-1, 1 mark]**

(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.

36. Consider the following pseudo code. What is the total number of multiplications to be performed? **[2014, Set-1, 2 Marks]**

```
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.

37. 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? **[2014, Set-2, 1 Mark]**

(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) $p = n * (n-1) * (n-2) / 6.0;$

38. 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**? **[2014, Set-2, 2 Marks]**

(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.

39. Consider the function func shown below:

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

The value returned by func (435) is __________. **[2014, Set-2, 1 Mark]**

40. 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);
}
```

Give a value q (to 2 decimals) such that f(q) will return q:_____. **[2014, Set-2, 2 Marks]**

41. The minimum number of arithmetic operations required to evaluate the polynomial $P(X) = X^5 + 4X^3 + 6X + 5$ for a given value of X, using only one temporary variable is _____. **[2014, Set-3, 1 Mark]**

42. Consider the following function:

```
int unknown (int n) {
    int i, j, k=0;
    for (i=n/2; i<=n; i++)
    for (j=2; j<=n; j=j*2)
        k = k + n/2;
    return (k);
}
```

The return value of the function is **[2013, 2 Marks]**

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$ (d) $\Theta(n^3 \log n)$

43. What is the return value of `f (p,p)`, if the value of `p` is initialized to 5 before the call? Note that the first parameter is passed by reference, whereas the second parameter is passed by value. **[2013, 2 Marks]**

```
int f (int &x, int c) {
    c = c – 1;
    if (c==0) return 1;
    x = x + 1;
    return f(x,c) * x;
}
```

(a) 3024 (b) 6561

(c) 55440 (d) 161051

44. Consider the program given below, in a block-structured pseudo-language with lexical scoping an nesting of procedures permitted.

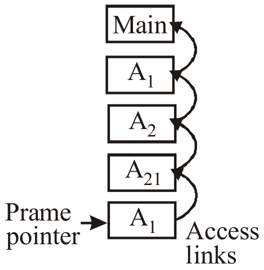
```
Program main;
 Var ......
  Procedure A1;
        Var .....
        Call A2;
  End A1
  Procedure A2;
        Var ...
        Procedure A21;
           Var....
           Call A1;
        End A21
        Call A21;
  End A2;
  Call A1;
End main.
```

Consider the calling chain:

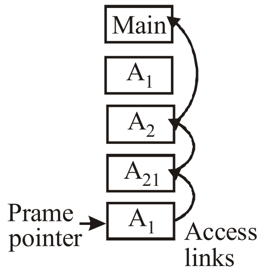
Main → A1 → A2 → A21 → A1

The correct set of activation records along with their access links is given by **[2012, 2 Marks]**

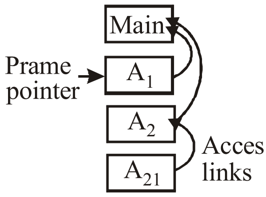
(a)



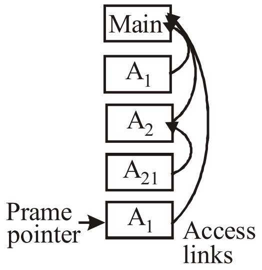
(b)



(c)



(d)



45. What will be the output of the following C program segment? **[2012, 2 Marks]**

```
char inChar = 'A';
switch (inChar) {
case 'A' : prinf("Choice A\n");
case 'B' :
case 'C' : printf ("Choice B'):
case 'D' :
case 'E' :
default : printf ("No Choice");}
```

(a) No choice

(b) Choice A

(c) Choice A choice B No choice

(d) Program gives no output as it is erroneous

46. What does the following program print? **[2011, 2 Marks]**

```
#include < stdio.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);
}
```

(a) 2 2 (b) 2 1

(c) 0 1 (d) 0 2

47. What does the following fragment of C-program print? **[2011, 1 mark]**

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

(a) GATE 2011 (b) E 2011

(c) 2011 (d) 011

Common Data for Questions 48 and 49.

Consider the following recursive C function that takes two arguments unsigned into foo (unsigned int, n, unsigned int r) {
if n > 0 return n% foo (n/r, r);
else return 0,
}

48. What is the return value of the function foo, when it is called as foo (513, 2)? **[2011, 2 Marks]**

(a) 9 (b) 8

(c) 5 (d) 2

49. What is the return value of the function foo, when it is called as foo (345, 10)? **[2011, 2 Marks]**

(a) 345 (b) 12

(c) 5 (d) 3

50. The following program is to be tested for statement coverage. **[2010, 2 Marks]**

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

The test cases T_1, T_2, T_3 and T_4 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.

T_1: a, b, c and d are all equal

T_2: a, b, c and d are all distinct

T_3: a = b and c! = d

T_4: a! = b and c = d

Which of the test suites given below ensures coverage of statements S_1, S_2, S_3 and S_4?

(a) T_1, T_2, T_3 (b) T_2, T_4

(c) T_3, T_4 (d) T_1, T_2, T_4

51. The program below uses six temporary variables a, b, c, d, e, f. **[2010, 2 Marks]**

```
a = 1
b = 10
c = 20
d = a + b
e = c + d
f = c + e
b = c + e
e = b + f
d = 5 + e
return d + f
```

Assuming that all operations take their operands from registers, what is the minimum number of registers needed to execute this program without spilling?

(a) 2 (b) 3

(c) 4 (d) 6

52. What is the value printed by the following C program? **[2010, 2 Marks]**

```
#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};
    print f (“%d, f(a, 6));
    return 0,
}
```

(a) – 9 (b) 5

(c) 15 (d) 19

53. Consider the program below: **[2009, 1 Mark]**

```
#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

54. Which combination of the integer variables x, y and z makes the variable a get the value 4 in the following expression?

a = (x > y)? ((x > z)? x : z) : ((y > z)? y : z) **[2008, 1 Mark]**

(a) x = 3, y = 4, z = 2

(b) x = 6, y = 5, z = 3

(c) x = 6, y = 3, z = 5

(d) x = 5, y = 4, z = 5

55. What is printed by the following C program? **[2008, 2 Marks]**

```
int f (int x, int * py, int ** ppz)
{
        int y, z;
        **ppz + = 1; z = *ppz;
        *py + = 2; y = *py;
        x + = 3;
        return x + y + z;
}
void main ()
{
        int c, *b, **a,
        c = 4; b & c; a = & b
        printf(“%d”, f(c, b, a));
}
```

(a) 18 (b) 19

(c) 21 (d) 22

56. 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. **[2008, 2 Marks]**

```
void recerse (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);

57. Which of the following are true? **[2008, 2 Marks]**

1. A programming language which does not permit global variables of any kind and has no nesting of procedures/functions, but permits recursive can be implemented with static storage allocation
2. Multi-level access link (or display) arrangement is needed to arrange activation records only, if the programming language being implemented has nesting of procedures/functions.
3. Recursion in programming languages cannot be implemented with dynamic storage allocation
4. Nesting of procedures/functions and recursion require a dynamic heap allocation scheme and cannot be implemented with a stack-based allocation scheme for activation records.
5. Programming languages which permit a function to return a function as its result cannot be implemented with a stack based storage allocation scheme for activation records

(a) 2 and 5 (b) 1, 3 and 4
(c) 1, 2 and 5 (d) 2, 3 and 5

58. Consider the following segment of C-code **[2007, 1 Mark]**

```
int j, n;
      j = 1;
      while (j < = n)
                          j = j*2;
```

The number of comparisons made in the execution of the loop for any n > 0 is

(a) $\lceil \log_2 n \rceil + 1$ (b) n
(c) $\lceil \log_2 n \rceil$ (d) $\lfloor \log_2 n \rfloor + 1$

59. Consider the following C function: **[2007, 2 Marks]**

```
      int f (int n)
      {static int r = 0;
      if (n < = 0) return 1;
        if (n > 3)
          {r = n;
         return f (n – 2) + 2;
          }
      return f(n – 1) + r;
}
```

What is the value of f(5)?
(a) 5 (b) 7
(c) 9 (d) 18

60. Consider this code to swap integers and these five statements:
The code **[2006, 2 Marks]**

```
void swap (int *px, int *py) {
      *px = *px – *py;
     *py = *px + *py;
     *px = *py – *px;
   }
```

S1 : will generate a compilation error
S2 : may generate a segmentation fault by runtime depending on the arguments passed
S3 : correctly implements the swap procedure for all input pointers referring to integers stored in memory locations accessible to the process
S4: implements the swap procedure correctly for some but not all valid input pointers
S5 : may add or subtract integers and pointers
(a) S1 only (b) S2 and S3
(c) S2 and S4 (d) S2 and S5

61. Consider these two functions and two statements S1 and S2 about them. **[2006, 2 Marks]**

```
int work1 (int*a, int i, int j)
{
      int x = a[i + 2];
       a[j] = x + 1;
      return a[i + 2] – 3;
}
```

```
int work2 {int *a, int i, int j)
{
      int t1 = i + 2
      int t2 = a[t1]
      a[j] = t2 + 1;
      return t2 – 3;
 }
```

S1: The transformation form work1 to work2 is valid, i.e., for any program state and input arguments, work2 will compute the same output and have the same effect on program state as work 1.

S2: All the transformations applied to work1 to get work2 will always improve the performance (i.e., reduce CPU time) of work2 compared to work1

(a) S1 is false and S2 is false
(b) S1 is false and S2 is true
(c) S1 is true and S2 is false
(d) S1 is true and S2 is true

62. Consider the following code written in a pass-by-reference language like FORTRAN and these statements about the code

```
      Subroutine swap (ix, iy)
                     it = ix
      L1 :           ix = iy
      L2 :           iy = it
            end
```

```
ia = 3
ib = 8
call swap (ia, ib + 5)
print *, ia, ib
end
```

S_1 : The compiler will generate code to allocate a temporary nameless cell, initialize it to 13 and pass the address of the cell swap

S_2 : On execution the code will generate a run time error on line L_1

S_3 : On execution the code will generate a run time error on line L_2

S_4 : The program will print 13 and 8

S_5 : The program will print 13 and –2

Exactly which of the following sets of statements is / are correct? **[2006, 2 Marks]**

(a) S_1 and S_2 (b) S_1 and S_4
(c) S_3 (d) S_1 and S_5

63. Which one of the following are essential features of an object-oriented programming language? **[2005, 1 Mark]**

1. Abstraction and encapsulation
2. Strictly - typedness
3. Type-safe property coupled with sub-type rule
4. Polymorphism in the presence of inheritance

(a) 1 and 2 (b) 1 and 4
(c) 1, 2 and 4 (d) 1, 3 and 4

64. A common property of logic programming languages and functional languages is **[2005, 1 Mark]**

(a) both are procedural languages
(b) both are based on λ-calculus
(c) both are declarative
(d) both use Horn-clauses

65. An Abstract Data Type (ADT) is **[2005, 1 Mark]**

(a) same as an abstract class
(b) a data type that cannot be instantiated
(c) a data type for which only the operations defined on it can be used, but none else
(d) All of the above

66. What does the following C-statement declare?
int (*f) int*); **[2005, 1 Mark]**

(a) A function that takes an integer pointer as argument and returns an integer
(b) A function that takes as argument and returns an integer pointer
(c) A pointer to a function that takes an integer pointer as argument and returns an integer
(d) A function that takes an integer pointer as argument and returns a function pointer

67. Match the following List I with List II and select the correct answer using the codes given below the lists.

List I		List II	
P.	Functional	1.	Command-based, procedural
Q.	Logic	2.	Imperative, abstract data types
R.	Object-oriented	3.	Side-effect free, declarative, expression evaluation
S.	Imperative	4.	Declarative, clausal representation, theorem proving

[2005, 2 Marks]

(a) P–2, Q–3, R–4, S–1 (b) P–4, Q–3, R–2, S–1
(c) P–3, Q–4, R–1, S–2 (d) P–3, Q–4, R–2, S–1

68. Consider the following C program segment: **[2005, 2 Marks]**

```
char p [20]
char *s = “string”;
int length = strlen (s);
for (i = 0; i < length, i++)
  p[i] = s [length – i];
print f(“%s”, p);
```

The output of the program is

(a) gnirts
(b) garbage dashing or no value printed
(c) gnirt
(d) no output is printed

69. Consider the following C-program **[2005, 2 Marks]**

```
double foo (double);            /* Line 1 */
int main () {
        double da, db;
        // input da
        db = foo (da);
}
        double foo (double a) {
        return a;
```

The above code compiled without any error or warning. If Line 1 is deleted, the above code will show

(a) no compile warning or error
(b) some compiler-warnings not leading to unintended results
(c) some compiler-warnings due to type-mismatch eventually leading to unintended results
(d) compiler errors

70. Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let $n = d_1 d_2 \ldots d_m$.

```
int n, rev;
rev = 0;
while (n > 0) {
      rev = rev * 10 + n% 10;
      n = n/ 10;
      }
```

The loop invariant condition at the end of the i^{th} iteration is **[2005, 2 Marks]**

(a) $n = d_1 d_2 \ldots d_{m-i}$ and rev $= d_m d_{m-1} \ldots d_{m-j+1}$
(b) $n = d_{m-i+1} \ldots d_{m-1}\ d_m$ or rev $= d_{m-\mathrm{i}} \ldots d_2 d_1$
(c) $n \neq$ rev
(d) $n = d_1 d_2 \ldots d_m$ or rev $= d_m \ldots d_2 d_1$

71. Consider the following C program :

```
main ()
      { int x, y, m, n;
      scanf (“%d %d”, &x, &y);
      / * Assume x > 0 and y > 0 */
      m = x; n = y;
      while (m! = n)
            {     if (m > n)
                  m = m – n;
               else
                  n = n – m;
            }
      printf (“ % d”, n);
      }
```

The program computes **[2005, 2 Marks]**

(a) $x + y$ using repeated subtraction
(b) x mod y using repeated subtraction
(c) the greatest common divisor of x and y
(d) the least common multiple of x and y

72. What does the following algorithm approximate? (Assume $m > 1, \in\ > 0$). **[2005, 2 Marks]**

```
x = m;
y = 1;
while (x– y > ∈ )
    {   x = (x + y) / 2;
        y = m/x;
    }
print (x);
```

(a) $\log m$ (b) m^2
(c) $m^{\frac{1}{2}}$ (d) $m^{\frac{1}{3}}$

73. Consider the following C function: **[2005, 2 Marks]**

```
int f (int n)
{ static int i = 1;
   if (n > = 5) return n;
  n = n + 1;
  i ++;
 return f(n);
  }
```

The value returned by f (1) is

(a) 5   (b) 6
(c) 7   (d) 8

74. Consider the following C-program: **[2005, 2 Marks]**

```
    void foo (int n, int sum) {
      int k = 0,  j = 0
      if (n = = 0) return;
      k = n% 10, j = n/10;
      sum = sum + k;
      foo (j, sum);
     printf (“%d”, k);
}
int main () {
    int a = 2048, sum = 0;
    foo (a, sum);
    printf (“%d\n”, sum);
```

What does the above program print?

(a) 8, 4, 0, 2, 14   (b) 8, 4, 0, 2, 0
(b) 2, 0, 4, 8, 14   (d) 2, 0, 4, 8, 0

75. The goal of structured programming is to **[2004, 1 Mark]**

(a) have well indented programmes
(b) be able to infer the flow of control from the compiled code
(c) be able to infer the flow of control form the program text
(d) avoid the use of GOTO statements

76. Consider the following C function :

```
void swap (int a, int b)
{int temp;
temp = a;
a = b;
b = temp;
}
```

In order to exchange the values of two variables x and y. **[2004, 1 Mark]**

(a) Call swap (x, y)
(b) Call swap (& x, & y)
(c) Swap (x, y) cannot be used as it does not return any value
(d) Swap (x, y) cannot be used as the parameters are passed by value

77. The best data structure to check whether an arithmetic expression has balanced parenthesis is a **[2004, 1 Mark]**

(a) queue   (b) stack
(c) tree   (d) list

78. Consider the following C function :

```
float f (float x, int y) {
      float p, s; int i;
for (s = 1), p = 1, i = 1; i < y ; i ++){
p* = x/i;
s + = p;
      }
      return s;
}
```

For large value of y, the return value of the function f best approximates **[2003, 1 Mark]**

(a) x^y (b) e^x
(c) $\ell n(1+x)$ (d) x^x

79. Which of the following statements is false? **[2003, 1 mark]**

(a) In statically types languages, each variable in program has a fixed type
(b) In un-typed languages, values do not have any types
(c) In dynamically typed languages, variable have no types
(d) In all statically typed languages, each variable in program is associated with value of only a single type during the execution of the program

80. Which of the following is not an advantage of using shared, dynamically linked libraries as opposed to using statically linked libraries? **[2003, 2 Marks]**

(a) Smaller sizes of executable files
(b) Lesser overall page fault rate in the system
(c) Faster program startup
(d) Existing programs need not be relinked to take advantage of newer versions

81. Consider the following class definition in a hypothetical object oriented language that supports inheritance and uses dynamic binding. The language should not be assuemed to be either Java or C++ , though the syntax is similar. **[2003, 2 Marks]**

```
Class P   {                  Class Q subclass of P {
void f (int i) {             void f (int i) {
print (i);                          print (2*i);
      }
}
```

Now, consider the following program fragment”

```
Px = new Q()
Qy = new Q();
Pz = new Q();
X.f(1); ((P)y). f(1); z.f(1);
```

Here, ((P)y) denotes a type cast of y to P. The output produced by executing the above program fragment will be

(a) 1 2 1   (b) 2 1 1
(c) 2 1 2   (d) 2 2 2

Common Data for Questions 82 and 83

The following program fragment is written in a progrmming language that allows global variable and does not allow nested declrations of functions.

```
global int i = 100, j = 5;
void P (x) {
        int i = 10;
        print (x + 10);
        i = 200;
        j = 20;
        print (x);
}
main () {P (i + j);}
```

82. If the programming language uses static scoping and call by need parameter passing mechanism, the values printed by the above program are **[2003, 2 Marks]**

(a) 115, 220 (b) 25,220

(c) 25,15 (d) 115, 105

83. If the programme language uses dynamic scoping and call by name parameter passing mechanism, the values printed by the above program are **[2003, 2 Marks]**

(a) 115, 220 (b) 25, 220

(c) 25, 15 (d) 115, 105

84. Consider the C program shown below :

```
        # include < stdio. h >
        # define print(x) printf (“ %d”, x)
        int x ;
        void Q (int z) {
        z + = x; print(z);
}
void P (int *y) {
        int x = *y + 2;
        Q (x); *y = x – 1;
        print (x);
}
main (void) {
        x = 5;
        P (& x)
        Print (x);
}
```

The output of this program is **[2003, 2 Marks]**

(a) 12 7 6 (b) 22 12 11

(c) 14 6 6 (d) 7 6 6

85. The results returned by function under value – result and reference parameter passing conventions **[2002, 1 Mark]**

(a) do not differ

(b) differ in the presence of loops

(c) differ in all cases

(d) may differ in the presence of exception

86. In the C language **[2002, 1 Mark]**

(a) at most one activation record exists between the current activation record and the activation record for the main

(b) the number of activation records between the current activation record and the activation record for the main depends on the actual function calling sequence

(c) the visibility of global variables depends on the actual function calling sequence

(d) recursion requires the activation record for the recursive function to be saved on a different stack before the recursive function can be called

87. What is printed by the print statements in the program P1 assuming call by reference parameter passing?

```
Program P1()
{
x = 10;
y = 3;
func1(y, x, x);
print x;
print y;
}
func 1 (x, y, z)
{
        y = y + 4;
        z = x + y + z;
}
```

[2002, 2 Marks]

(a) 10, 3 (b) 31, 3

(c) 27, 7 (d) None of these

88. Consider the following three C functions

```
[P1]    int*g(void)
        {
                intx = 10;
                return (&x);
        }
[P2]    int*g(void)
        {
                int* px;
                *px = 10;
                return px;
        }
[P3]    int*g(void)
        {
                int*px
                px = (int*) malloc (size of (int));
                *px = 10;
                return px;
        }
```

Which of the above 3 functions are likely to cause problems with pointers? **[2002, 2 Marks]**

(a) Only *P*3 (b) *P*1 and *P*3

(c) *P*1and *P*2 (d) *P*1, *P*2 and *P*3

89. Consider the following program :

```
        Program P2
                var n:int :
                procedure W (var x:int)
                begin
                x = x + 1;
                print x;
                end
                procedure D
                begin
                        varn:int;
                        n = 3;
                        W (n);
        End
begin                   || begin P2
        n = 10;
        D;
end
```

If the language has dynamic scopping and parameters are passed by reference, what will be printed by the program? **[2002, 2 Marks]**

(a) 10 (b) 11
(c) 3 (d) None of these

90. The process of assigning load addresses to the various parts of the program and adjusting the code and date in the program to reflect the assigned addresses is called **[2001, 1 Mark]**

(a) assembly (b) parsing
(c) relocation (d) symbol resolution

91. The most approximate matching for the following pairs

X. m = malloc (5); m = NULL; 1. using dangling pointers
Y. free (*n*); n- > value = 5; 2. using uninitialized pointers
Z. char *p; *p = 'a'; 3. lost memory

is **[2000, 1 Mark]**

(a) X – 1, Y – 3, Z – 2 (b) X – 2, Y – 1, Z – 3
(c) X – 3, Y – 2, Z – 1 (d) X – 3, Y – 1, Z – 2

92. The most appropriate matching for the following pairs is

X. depth first search 1. heap
Y. breadth-first search 2. queue
Z. sorting 3. stack

[2000, 1 Mark]

(a) X – 1, Y – 2, Z – 3 (b) X – 3, Y – 1, Z – 2
(c) X – 3, Y – 2, Z – 1 (d) X – 2, Y – 3, Z – 1

93. Aliasing in the context of programming languages refers to **[2000, 1 Mark]**

(a) multiple variables having the same memory location
(b) multiple variables having the same value
(c) multiple variables having the same identifier
(d) multiple uses of the same variables

94. Consider the following C declaration

```
struct {
     short s [ 5 ]
     union {
     float y ;
     long z ;
     } u;
} t;
```

Assume that objects of the type short, float and long occupy 2 byte, 4 byte and 8 byte, respectively. The memory requirement for variable *t*, ignoring alignment considerations, is **[2000, 1 Mark]**

(a) 22 byte (b) 14 byte
(c) 18 byte (d) 10 byte

95. The number of tokens in the following C statements print f("i = % d, & i = % x", i & i) is **[2000, 2 Marks]**

(a) 3 (b) 26
(c) 10 (d) 21

96. The value of *j* at the end of the execution of the following C program int incr (int *i*)

```
{
     static int count = 0;
     count = count + i;
     return (count);
}
main () {
     int i j;
     for (i = 0; i < = 4; i + +)
          j = incr (i);
}
```

is **[2000, 2 Mark]**

(a) 10 (b) 4
(c) 6 (d) 7

Arrays

97. Let *A* be an array of 31 numbers consisting of a sequence of 0's followed by a sequence of l's. The problem is to find the smallest index *i* such that *A[i]* is 1 by probing the minimum number of locations in *A*. The *worst case* number of probes performed by an *optimal* algorithm is _____. **[2017, Set 1, 2 Marks]**

98. Consider the following snippet of a C program. Assume that swap (&x, &y) exchanges the contents of x and y. **[2017, Set 2, 2 Marks]**

```
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>=l ; 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 _______.

99. A Young tableau is a 2D array of integers increasing from left to right and from top to bottom. Any unfilled entries are marked with ∞, and hence there cannot be any entry to the right of, or below a ∞. The following Young tableau consists of unique entries.

1	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

When an element is removed from a Young tableau, other elements should be moved into its place so that the resulting table is still a Young tableau (unfilled entries may be filled in with a ∞). The minimum number of entries (other than 1) to be shifted, to remove 1 from the given Young tableau is ________. **[2015, Set 2, 2 Marks]**

100. Suppose $C = \langle c[0], ..., c[k-] \rangle$ 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)
$z \leftarrow 1$
for $i \leftarrow 0$ to $k - 1$
do $z \leftarrow z^2$ mod n
if c[i] = 1
    then $z \leftarrow (z \times a)$ mod n
rutrun z
```

If k = 4, c = $\langle 1, 0, 1, 1 \rangle$, a = 2 and n = 8, then the output of DOSOMETHING (c, a, n) is _________.

[2015, Set 3, 2 Marks]

101. 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", ptr-p, **ptr);
}
```

The output of the program is ______.

[2015, Set 3, 2 Marks]

102. Consider the following two C code segments. Y and X are one and two dimensional arrays of size n and n × 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 segment 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

[2015, Set 3, 2 Marks]

(a) Only S2 is correct
(b) Only S3 is correct
(c) Only S1 and S2 are correct
(d) Only S1 and S3 are correct

103. 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

[2014, Set-1, 2 Marks]

(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**.

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

[2014, Set-3, 1 Mark]

```
    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

Common Data for Questions 105 and 106:

The procedure given below is required to find and replace certain characters inside an input character string supplied in array A. The characters to be replaced are supplied in array oldc, while their respective replacement characters are supplied in array newc. Array A has a fixed length of five characters, while arrays oldc and newc contain three characters each. However, the procedure is flawed.

```
void find_and_replace (char *A, char *oldc, char *newc)
{
      for (int i=0; i<5; i++)
            for (int j=0; j<3; j++)
            if (A[i]==oldc[j]) A[i] = newc[j];
}
```

The procedure is tested with the following four test cases.

1. oldc = “abc”, newc = “da b”

2. oldc = “cde”, newc = “bcd”

3. oldc = “bca”, newc = “cda”

4. oldc = “abc”, newc = “bac”

105. The tester now tests the program on all input strings of length five consisting of characters ‘a’, ‘b’, ‘c’, ‘d’ and ‘e’ with duplicates allowed. If the tester carries out this testing with the four test cases given above, how many test cases will be able to capture the flaw? **[2013, 2 Marks]**

(a) Only 1 (b) Only 2

(c) Only 3 (d) All four

106. If array A is made to hold the string “abcde”, which of the above four test cases will be successful in exposing the flaw in this procedure? **[2013, 2 Marks]**

(a) 2 only (b) 4 only

(c) 3 and 4 only (d) None

Statements for Linked Answer Questions 107 and 108

The subset - sum problem is defined as follows : Given a set of n positive integers, $S = \{a_1, a_2, a_3, ...a_n\}$, and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array, X, with n rows and $W + 1$ columns. $X[i, j]$, $1 \le i \le n, 0 \le j \le W$, is true if and only if there is a subset of $\{a_1, a_2, ...a_i\}$ whose elements sum to j

107. Which of the following is valid for $2 \le i \le n$ and $a_i \le j \le W$? **[2008, 2 Marks]**

(a) $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$

(b) $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$

(c) $X[i, j] = X[i-1, j] \wedge X[i,\ j-a_i]$

(d) $X[i, j] = X[i-1, j] \wedge X[i-1, j-a_i]$

108. Which entry of the array X, if true, implies that there is a subset whose elements sum to W? **[2008, 2 Marks]**

(a) $X[1, W]$ (b) $X[n, 0]$

(c) $X[n, W]$ (d) $X[n-1, n]$

Statements for linked Answer Questions 109 and 110

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 u, 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) print f (“x is in the array”);
9.         else print f (“x is not in the array”);
10.        }
```

109. On which of the following contents of Y and x does the program fail? **[2008, 2 Marks]**

(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

110. The correction needed in the program to make it work properly is **[2008, 2 Marks]**

(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))

111. An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers needs to be determined. Which of the following is true about the number of comparisons needed? **[2007, 2 Marks]**

(a) At least 2n – c comparisons, for some constant c, are needed

(b) At most 1.5n – 2 comparisons are needed

(c) At least $n\log_2 n$ comparisons are needed

(d) None of the above

112. A set X can be represented by an array x [n] as follows

$$x[i]\begin{cases} 1\ , & \text{if } i \in X \\ 0\ , & \text{otherwise} \end{cases}$$

Consider the following algorithm in which x, y and z are Boolean arrays of size n :

```
algorithm zzz (x[], y[], z []) {
int i;
for (i = 0; i < n; + + i )
      z [i] = (x[i] ^ ~ y [i] ∨ (~ x [i] ^ y [i])
      }
```

The set Z computed by the algorithm is **[2006, 2 Marks]**

(a) $(X \cup Y)$ (b) $(X \cap Y)$

(c) $(X - Y) \cap (Y - X)$ (d) $(X - Y) \cup (Y - X)$

113. Two matrices M_1 and M_2 are to be stored in arrays A and B respectively. Each array can be stored either in row–major or column–major order in contiguous memory locations. The time complexity of an algorithm to compute $M_1 \times M_2$ will be **[2004, 2 Marks]**

(a) best if A is in row – major, and B is in column major order

(b) best if both are in row – major order

(c) best if both are in column – major order

(d) independent of the storge scheme

114. Consider the following C-function in which a[n] and b[m] are two sorted integer arrays and c[n + m] be another array.

void xyz (int a [ ], int b [ ], int c [ ]) **[2006, 2 Marks]**

```
{
int i, j, k;
i = j = k = 0;
while ((i < n) && (j < m))
  if (a [i] < b [j] c [k++] = a[i++];
 else c[k++] = b[j++];
}
```

Which of the following conditions hold(s) after the termination of the while loop?

(i) j < m, k = n + j – 1 and a [n – 1] < b[j], if i = n

(ii) i < n, k = m + i – 1 and b [m – 1] ≤ a[i], if j = m

(a) only (i)

(b) only (ii)

(c) either (i) or (ii) but not both

(d) neither (i) nor (ii)

115. A program P reads in 500 integers in the range [0, 100] representing the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies? **[2005, 1 Mark]**

(a) An array of 50 numbers

(b) An array of 100 numbers

(c) An array of 500 numbers

(d) A dynamically allocated array of 550 numbers

116. A single array A [1...MAXSIZE] is used to implement two stacks. The two stack graw from opposite ends of the array. Variable top 1 and top 2 (top 1 < top 2) point to the location of the topmost element in the each of the stacks. If the space is to be used efficiently, the condition for stack full is **[2004, 1 Mark]**

(a) (top1 – MAXSIZE / 2) and (top 2 = MAXSIZE/2 + 1)

(b) top1 + top 2 = MAXSIZE

(c) (top1 = MAZSIZE/2) or (top 2 = MAXSIZE)

(d) top1 = top 2 – 1

117. Assume the following C variable declaration

int* A [10], B [10] [10];

Of the following expressions which will not give compile time errors if used as left hand sides of assignment statements in a C program ? **[2003, 1 Mark]**

(i) *A* [2](ii) *A* [2] [3]

(iii) *B* [1] (iv) *B* [2] [3]

(a) *I, II and IV only* (b) *II, III and IV only*

(c) *II, and IV only* (d) *IV only*

118. In the following C program fragment j, k n and Two Log_n are integer variables, and A is an array of integers. The variable *n* is initialized to an integer ≥ 3, and TwoLog_n is initialized to the value of $2 * \lceil \log_2(n) \rceil$

```
for (k = 3; k < = n; k + +)
      A [k] = 0;
for (k = 2; k < = TwoLog_n; k + +)
      for (j = k + 1; j < = n; j + +)
      A[j] = A[j] || (j%k);
      for (j = 3; j < = n; j + +)
          if (!A[j]) printf (“%d”, j);
```

The set of numbers printed by this program fragment is **[2003, 2 Marks]**

(a) $\{m \mid m \leq n \ (\exists i)\ [m = i!]\}$

(b) $\{m \mid m \leq n, \ (\exists i)\ [m = i^2]\}$

(c) $\{m \mid m \leq n$, (m is prime)$\}$

(d) { }

119. Following declaration of a two-dimensional array in C.

Char a [100] [100];

Assuming that the main memory is byte addressable and that the array is stored starting from memory address 0, the address of a [40] [50] is **[2002, 2 Marks]**

(a) 4040 (b) 4050

(c) 5040 (d) 5050

120. The following C declarations

```
struct node{
      int i :
      float j ;
}
struct node * s [10];
```

define s to be **[2000, 1 Mark]**

(a) an array, each element of which is a pointer to a structure of type node

(b) a structure of 2 fields, each field being a pointer to an array of 10 elements

(c) a structure of 3 fields : an integer, a float and an array of 10 elements

(d) an array, each element of which is a structure of type node

121. Suppose you are given an array s[1....*n*] and a procedure reverse (*s,i,j*) which reverses the order of elements in a between positions *i* and *j* (both inclusive). What does the following sequence do, where $1 \leq k \leq n$: **[2000, 2 Marks]**

reverse (*s*, 1, *k*);
reverse (s, *k* + 1, *n*);
reverse (*s*, 1, *n*);

(a) Rotates *s* left by *k* positions

(b) Leaves *s* unchanged

(c) Reverses all elements of *s*

(d) None of these

122. Suppose you are given arrays p[1 ...N] and q[1...N] both uninitialized that is, each location may contain an arbitrary value), and a variable count, initialized to 0. Consider the following procedures set and iset: **[2000, 5 Marks]**

```
Set (i) {
      count = count + 1;
      q [count] = i;
      p[i] = count;
}
is_set(i) {
      if (p[i] ≤ 0 or p[i] > count)
      return false;
      if (q[p[i]] ≠ i)
      return false;
return true;
}
```

(a) Suppose we make the following sequence of calls: set (7); set (3); set(9);
After these quence of calls, what is the value of count, and what do q[1], q[2], q[3], p[7], p[3] and p[9] contain?

(b) Complete the following statement “The first count elements of .... contain values i such that set (....................) has been called”.

(c) Show that if set (i) has not been called for some i, then regardless of what p[i] contains, is_set (i) will return false.

123. A recursive program to compute Fibonacci numbers is shown below. Assume you are also given an array f[0...m] with all elements initialized to 0. **[2000, 5 Marks]**

```
fib(n){
    if (n > m) error ();
    if (n = = 0) return 1;
    if (n = = 1) return 1;
    if (□) ......................... (1)
    return □ ......................... (2)
    t = fib (n – 1) + fib (n – 2);
    □ ......................... (3)
    return t;
}
```

(a) Fill in the boxes with expressions/statements to make fib() store and reuse computed Fibonacci values. Write the box number and the corresponding contents in your answer book.

(b) What is the time complexity of the resulting program when computing fib(n)?

124. An array contains four occurrences of 0, five occurrences of 1, and three occurrences of 2 in any order. The array is to be sorted using swap operations (elements that are swapped need to be adjacent). **[2000, 5 Marks]**

(a) What is the minimum number of swaps needed to sort such an array in the worst case?

(b) Give an ordering of elements in the above array so that the minimum number of swaps needed to sort the array is maximum.

Stacks

125. Which one of the following is TRUE at any valid state in shift – reduce parsing? **[2015, Set 1, 1 mark]**

(a) Viable prefixes appear only at the bottom of the stack and not inside

(b) Viable prefixes appear only at the top of the stack and not inside

(c) The stack contains only a set of viable prefixes

(d) The stack never contains viable prefixes

126. 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? **[2014, Set-2, 2 Marks]**

(a) A queue cannot be implemented using this stack.

(b) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a 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.

127. The following postfix expression with single digit operands is evaluated using a stack: **[2007, 2 Marks]**

8 2 3 ^ / 2 3 * + 5 1 * -

Note that ^ is the exponentiation operator. The top two elements of the stack after the first * is evaluated are

(a)	6, 1	(b)	5, 7
(c)	3, 2	(d)	1, 5

128. An implementation of a queue Q, using two stacks S1 and S2 is given below **[2006, 2 Marks]**

```
    void insert (Q, x) {
      push (S1, x);
    }
    void delete (Q) {
if (stack-empty (S2)) then
    if (stack-empty (S1)) then {
        print (“Q is empty”);
        return;
    }
      else while (! (stack-empty (S1))) {
    x = pop (S1);
    push (S2, x);
    }
x = pop (S2);
}
```

Let n insert and m(≤ n) delete operations be performed in an arbitrary order on an empty queue Q. Let x and y be the number of push and pop operations performed respectively in the process. Which one of the following is true for all m and n?

(a) $n + m \le x < 2n$ and $2m \le y \le n + m$

(b) $n + m \le x \le 2n$ and $2m \le y \le 2n$

(c) $2m \le x < 2n$ and $2m \le y \le n + m$

(d) $2m \le x < 2n$ and $2m \le y \le 2n$

129. Let *S* be a stack of size n ≥ 1. Starting with the empty stack, suppose we push the first *n* natural numbers in sequence, and then perform *n* pop operations. Assume that Push and Pop operations take *X* seconds each, and *Y* seconds elapse between the end of one such stack operations and the start of the next operation. For m ≥ 1, define the stack – life of *m* as the time elapsed from the end of Push (m) to the start of the pop operation that removes m from S. The average stack – life of an element of this stack is **[2003, 2 Marks]**

(a)	$n (X + Y)$	(b)	$3Y + 2X$
(c)	$n (X + Y) – x$	(d)	$Y + 2X$

130. To evaluate an expression without any embedded function calls **[2002, 2 marks]**

(a) one stack is enough

(b) two stacks are needed

(c) as many stacks as the height of the expression tree are needed

(d) a turning machine is needed in the general case

131. What is the minimum number of stacks of size n required to implement a queue of size n? **[2002, 2 Marks]**

(a)	one	(b)	Two
(c)	Three	(d)	Four

Queues

132. 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 *0* (1) time? **[2017, Set 2, 1 Mark]**

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

133. 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)? **[2016, Set 1, 1 Mark]**

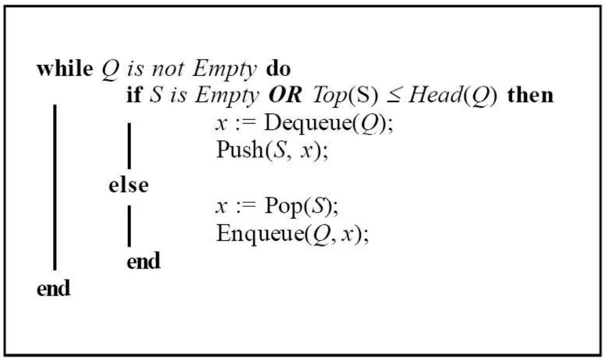
(a) Both operations can be performed in $O(1)$ time

(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 $\Omega(\log n)$

134. 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. **[2016, Set 1, 2 Mark]**



```
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 _________.

135. Consider the following operation along the Enqueue and Dequeue operations on queues, where k is a global parameter.

```
Multi-Dequeue(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 queue operations on an initially empty queue? **[2013, 2 Marks]**

(a) $\Theta(n)$ (b) $\Theta(n + k)$

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

136. 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 **[2012, 2 Marks]**

(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 = = REAR

(d) full (FRONT + 1) mod n = = REAR
empty: (REAR = = FRONT

137. A priority–queue is implemented as a max–heap. Initially, it has 5 elements. The level – order traversal of the heap is given below.
10, 8, 5, 3, 2
Two new elements 1 and 7 are inserted in the heap in that order. The level – order traversal of the heap after the insertion of the element is **[2005, 2 Marks]**

(a) 10, 8, 7, 5, 3, 2, 1 (b) 10, 8, 7, 2, 3, 1, 5

(c) 10, 8, 7, 1, 2, 3, 5 (d) 10, 8, 7, 3, 2, 1, 5

Linked Lists

138. *N* items are stored in a sorted doubly linked list. For a *delete* operation, a pointer is provided to the record to be deleted. For a *decrease-key* operation, a pointer is provided to the record on which the operation is to be performed.

An algorithm performs the following operations on the list in this order: $\Theta(N)$ *delete*, $O(\log N)$ *insert*, $O(\log N)$ find, and $\Theta(N)$ decrease-key. What is the time complexity of all these operations put together? **[2016, Set 2, 1 mark]**

(a) $O(\log^2 N)$ (b) $O(N)$

(c) $O(N^2)$ (d) $\Theta(N^2 \log N)$

139. The following C function takes a simply-linked list as input argument. It modifies the list as input argument. It modifies 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. **[2010, 2 Marks]**

```
type def 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;

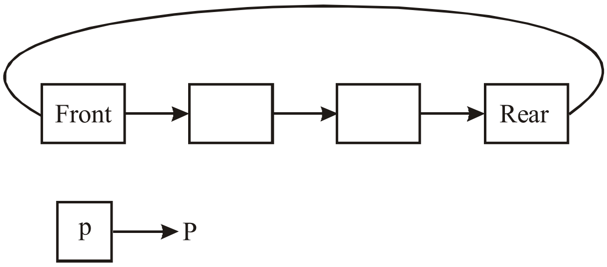
140. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution? **[2008, 2 Marks]**

```
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

141. A circularly linked list is used to represent a queue. A single variable p is used to access the queue. To which node should p point such that both the operations en-queue and de-queue can be performed in constant time? **[2005, 2 Marks]**



(a) Rear node
(b) Front node
(c) Not possible with a single pointer (d)
(d) Node next to front

142. Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership, cardinality will be the slowest? **[2004, 2 Marks]**
(a) Union only
(b) Intersection, membership
(c) Membership, cardinality
(d) Union, intersection

143. Consider the function f defined below :

```
    struct item {
        int data;
        struct item * next :
};
int f (struct item * p) {
    return ((p = = NULL) ||(p – > next = = NULL) ||
        ((p– > data < = p – > next –> data) &&
        f (p–> next)));
}
```

For a given linked list *p*, the function *f* returns 1, if and only, if **[2003, 2 Marks]**
(a) the list is empty or has exactly one element
(b) the elements in the list are sorted in non-decreasing order of data value
(c) the elements in the list are sorted in non-increasing order of data value
(d) not all element in the list have the same data value

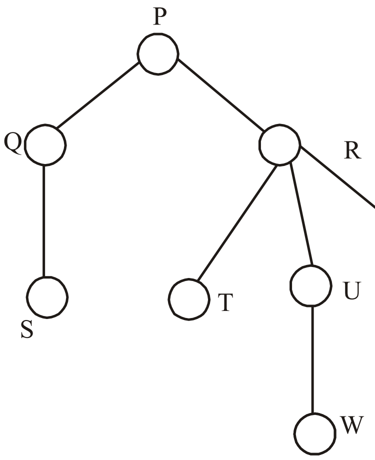
144. In the worst case, the number of comparisons needed to search a singly linked list of length *n* for a given element is **[2002, 1 Mark]**
(a) log n (b) $\frac{n}{2}$
(c) $\log_2{}^{(n-1)}$ (d) n

Trees

145. Let *T* be a tree with 10 vertices. The sum of the degrees of all the vertices in *T* is _______. **[2017, Set 1, 1 Mark]**

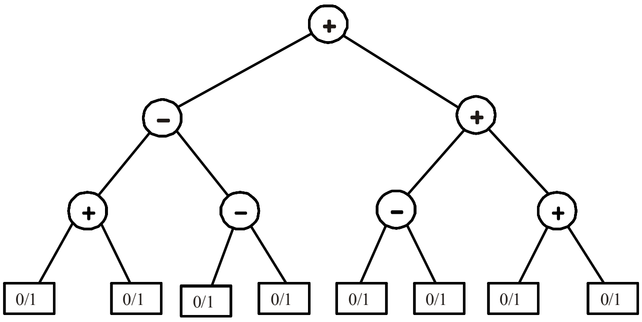
146. The result evaluating the postfix expression
10, 5, +, 60, 6, /, *, 8, – is **[2015, Set 3, 1 Mark]**
(a) 284 (b) 213
(c) 142 (d) 71

147. Consider the following rooted tree with the vertex labeled P as the root :



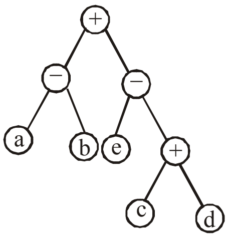
The order in which the nodes are visited during an in-order traversal of the tree is **[2014, Set-3, 1 Mark]**
(a) SQPTRWUV (b) SQPTUWRV
(c) SQPTWUVR (d) SQPTRUWV

148. Consider the expression tree shown. Each leaf represents a numerical value, which can either be 0 or 1. Over all possible choices of the values at the leaves, the maximum possible value of the expression represented by the tree is ______.



[2014, Set-2, 2 Marks]

149. Consider evaluating the following expression tree on a machine with load store architecture in which memory can be accessed only through load and store instructions. The variables a, b, c, d and e are initially stored in memory. The binary operators used in this expression tree can be evaluated by the machine only when the operands are in registers. The instructions produce result only in a register. If no intermediate results can be stored in memory, what is the minimum number of registers needed to evaluate this expression? **[2011, 2 Marks]**



(a) 2 (b) 9
(c) 5 (d) 3

150. 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. **[2009, 1 Mark]**

(a) 2 (b) 3
(c) 4 (d) 5

151. A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place? **[2008, 2 Marks]**

(a) 3 (b) 4
(c) 5 (d) 6

152. A complete n-array tree is a tree in which each node has n children or no children. Let l be the number of internal nodes and L be the number of leaves in a complete n-array tree. If L = 41, and l = 10, what is the value of n? **[2007, 2 Marks]**

(a) 3 (b) 4
(c) 5 (d) 6

153. In a complete k-array tree, every internal node has exactly k children. The number of leaves in such a tree with n internal nodes is **[2005, 2 Marks]**

(a) nk (b) (n – 1) k + 1
(c) n (k – 1) + 1 (d) n (k – 1)

154. Assume that the operators +, –, ×, are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, ×, +, –. The postfix expression corresponding to the infix expression a + b × c – d ^ e ^ f is **[2005, 2 Marks]**

(a) abc × + def ^^– (b) abc × + de^f^–
(c) ab + c × d – e^f^ (d) – + a × bc ^^ def

155. Consider the following C program segment **[2005, 2 Marks]**

```
struct Cell Node{
       struct Cell Node *left Child;
       int element;
       struct Cell Node *right Child;
       }
int Dosomething (struct Cell Node *ptr)
{
       int value = 0;
       if (ptr! = NULL)
              { if (ptr-> left Child! = NULL)
                     value = 1 + Dosomething (ptr -> left Child);
              if(ptr-> rightChild! = NULL)
              value = max (value, 1 + Dosomething (ptr->rightChild));
              }
       return (value);
}
```

The value returned by the function Dosomething when a pointer to the root of a non-empty tree is passed as argument is

(a) the number of leaf nodes in the tree
(b) the number of nodes in the tree
(c) the number of internal nodes in the tree
(d) the height of the tree

156. The number of leaf nodes in a rooted tree of n nodes, with each node having 0 or 3 children is **[2002, 2 Marks]**

(a) $\frac{n}{2}$ (b) $\frac{(n-1)}{3}$
(c) $\frac{(n-1)}{2}$ (d) $\frac{(2n+1)}{3}$

157. (a) Suppose you are given an empty B+-tree where each node (leaf and internal) can store up to 5 key values. Suppose values 1, 2, ... 10 are inserted, in order, into the tree, show the tree pictorially
(i) After 6 insertions, and
(ii) After all 10 insertions
Do NOT show intermediate stages.

(b) Suppose instead of splitting a node when it is full, we try to move a value to the left sibling. If there is no left sibling, or the left sibling is full, we split the node. Show the tree after values, 1, 2, ..., 9 have been inserted. Assume, as in (a) that each node can hold up to 5 keys.

(c) In general, suppose a B+-tree node can hold a maximum of m keys, and you insert a long sequence of keys in increasing order. Then what approximately is the average number of keys in each leaf level node.
(i) In the normal case, and
(ii) with the insertion as in (b). **[2000, 5 Marks]**

Binary Tree and Binary Search Trees

158. Let T be a binary search tree with 15 nodes. The minimum and maximum possible heights of T are: **[2017, Set 1, 1 Mark]**

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

159. 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: **[2017, Set 2, 2 Marks]**

(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

160. In a B^+ tree, if the search-key value is 8 bytes long, the block size is 512 bytes and the block pointer size is 2 bytes, then the maximum order of the B^+ tree is ______. **[2017, Set 2, 2 Marks]**

161. B+ Trees are considered **BALANCED** because **[2016, Set 2, 1 Mark]**

(a) the lengths of the paths from the root to all leaf nodes are all equal.

(b) the lengths of the paths from the root to all leaf nodes differ from each other by at most 1.

(c) the number of children of any two non-leaf sibling nodes differ by at most 1.

(d) the number of records in any two leaf nodes differ by at most 1.

162. Consider the following New-order strategy for traversing a binary tree: **[2016, Set 2, 2 Marks]**

- Visit the root;
- Visit the right subtree using New-order;
- Visit the left subtree using New-order;

The New-order traversal of the expression tree corresponding to the reverse polish expression 3 4 * 5 – 2 ^ 6 7 * 1 + – is given by:

(a) + – 1 6 7 * 2 ^ 5 – 3 4 *

(b) – + 1 * 6 7 ^ 2 – 5 * 3 4

(c) – + 1 * 7 6 ^ 2 – 5 * 4 3

(d) 1 7 6 * + 2 5 4 3 * – ^ –

163. 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 ________. **[2016, Set 2, 2 Marks]**

Note: The height of a tree with a single node is 0.

164. 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 **[2015, Set 1, 1 Mark]**

(a) 63 and 6, respectively

(b) 64 and 5, respectively

(c) 32 and 6, respectively

(d) 31 and 5, respectively

165. Which of the following is/are correct inorder traversal sequence(s) of binary search tree(s)?

I. 3, 5, 7, 8, 15, 19, 25 II. 5, 8, 9, 12, 10, 15, 25

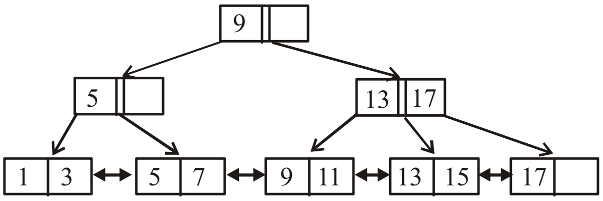
III. 2, 7, 10, 8, 14, 16, 20 IV. 4, 6, 7, 9, 18, 20, 25

[2015, Set 1, 1 Mark]

(a) I and IV only (b) II and iii only

(c) II and IV only (d) II only

166. With reference to the B^+ tree index of order 1 shown below, the minimum number of nodes (including the Root node) that must be fetched in order to satisfy the following query: "Get all records a search key greater than or equal to 7 and less than 15" is ____. **[2015, Set 2, 1 mark]**



167. Consider a complete binary tree where the left and the right subtrees of the root are max-heaps. The lower bound for the number operations to convert the tree to a heap is **[2015, Set 2, 1 Mark]**

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

(c) $\Omega(n \log n)$ (d) $\Omega(n^2)$

168. A binary tree T has 20 leaves. The number of nodes in T having two children is _____. **[2015, Set 2, 1 Mark]**

169. 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 **[2015, Set 3, 1 Mark]**

(a) 65 (b) 67

(c) 69 (d) 83

170. Consider a binary tree T that has 200 leaf nodes. Then, the number of nodes in T that have exactly two children are _______. **[2015, Set 3, 1 Mark]**

171. If the following system has non – trivial solution

$px + qy + rz = 0$

$qx + ry + pz = 0$

$rx + py + qz = 0$

then which one of the following options is TRUE?

(a) $p - q + r = 0$ or $p = q = -r$

(b) $p + q - r = 0$ or $p = -q = r$

(c) $p + q + r = 0$ or $p = q = r$

(d) $p - q + r = 0$ or $p = -q = -r$

172. Consider a B+ tree in which the search key is 12 bytes long, block size is 1024 bytes, record pointer is 10 bytes long and block pointer is 8 bytes long. The maximum number of keys that can be accommodated in each non – leaf node of the tree is ______. **[2015, Set 3, 2 Marks]**

173. 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 ______. **[2014, Set-1, 1 Mark]**

174. Suppose we have a balanced binary search tree *T* holding *n* numbers. We are given two numbers *L* and H and wish to sum up all the numbers in *T* that lie between *L* and *H*. Suppose there are m such numbers in *T*. If the tightest upper bound on the time to compute the sum is $O(n^a \log^b n + m^c \log^d n)$, the value of $10b + 100c + 1000d$ is _______. **[2014, Set-3, 2 Marks]**

175. Which one of the following is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of n nodes? **[2013, 1 Mark]**

(a) O(1) (b) O(log n)

(c) O(n) (d) O(n log n)

176. The pre-order traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the post-order traversal sequence of the same tree? **[2013, 2 Marks]**

(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

177. The height of a tree is defined as the number of edges on the longest path in the tree. The function shown in the below in invoked as height (root) to compute the height of a binary tree rooted at the tree pointer root. **[2012, 2 Marks]**

```
{    if (n = = NULL) return – 1;
     if (n → left = = NULL)
       if (n → right = = NULL) return 0;

       else return $\boxed{B1}$;        // Box 1

else {h1 = height (n → left);
     if (n → right = NULL) return (1 + h1);
     else {h2 = height (n → right);

          return $\boxed{B2}$;          // Box 1
          }
       }
}
```

The appropriate expressions 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)

178. We are given a set of n distinct elements and an unlabelled binary tree with n nodes. In how ways can we populate the tree with the given set so that it becomes a binary search tree? **[2011, 2 Marks]**

(a) 0 (b) 1

(c) n! (d) $\frac{1}{n+1}\,{}^{2n}C_n$

179. In a binary tree with n nodes, every node has an odd number of descendant. Every node is considered to be its own descendant. What is the number of nodes in the tree that have exactly one child? **[2010, 1 Mark]**

(a) 0 (b) 1

(c) (n – 1)/2 (d) n – 1

180. 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? **[2008, 2 Marks]**

(a) Θ (log n)

(b) Θ (n)

(c) Θ (n log n)

(d) None of the above, as the tree cannot be uniquely determined

181. A scheme for storing binary trees in an array X is as follows. Indexing of X starts at 1 instead of 0. The root is stored at X[1].For a node stored at X[i], the left child, if any, is stored in X[2i] and the right child, if any in X[2i | 1]. To be able to store any binary tree on n vertices the minimum size of X should be **[2007, 2 Marks]**

(a) $\log_2 n$ (b) n

(c) 2n + 1 (d) $2^n - 1$

182. The maximum number of binary trees that can be formed with three unlabelled nodes is **[2007, 1 Mark]**

(a) 1 (b) 5

(c) 4 (d) 3

183. The height of a binary tree is the maximum number of edges in any root to leaf path. The maximum number of nodes in a binary tree of height h is **[2007, 1 Mark]**

(a) $2^h - 1$ (b) $2^{h-1} - 1$

(c) $2^{h+1} - 1$ (d) 2^{h+1}

184. The inorder and preorder traversal of a binary tree are d b e a f c g and a b d e c f g, respectively The postorder traversal of the binary tree is. **[2007, 2 Marks]**

(a) d e b f g c a (b) e d b g f c a

(c) e d b f g c a (d) d e f g b c a

185. Consider the following C program segment where CellNode represents a node in a binary tree: **[2007, 2 Marks]**

```
struct Cell Node {
        struct Cell Node *leftChild;
          int element;
        struct CellNode *rightChild;
}
int GetValue (struct CellNode *ptr) {
int value = 0
if (ptr ! = NULL)
        if ((ptr - > leftChild = = NULL) &&
          (ptr -> rightChild == NULL){
        value = 1;
else
        value = value + GetValue (ptr -> leftChild)
                    + GetValue (ptr -> rightChild);
      }
return (value);
}
```

The value returned by GetValue when a pointer to the root of a binary tree is passed as its argument is

(a) the number of nodes in the tree

(b) the number of internal nodes in the tree

(c) the number of leaf nodes in the tree

(d) the height of the tree

186. The inorder and preorder traversal of a binary tree are d b e a f c g and a b d e c f g, respectively
The postorder traversal of the binary tree is **[2007, 2 Marks]**

(a) d e b f g c a (b) e d b g f c a

(c) e d b f g c a (d) d e f g b c a

187. Postorder traversal of a given binary search tree, T produces the following sequence of keys

10, 9, 23, 22, 27, 25, 15, 50, 95, 60, 40, 29

Which one of the following sequences of keys can be the result of an in-order traversal of the tree T? **[2005, 2 Marks]**

(a) 9, 10, 15, 22, 23, 25, 27, 29, 40, 50, 60, 95

(b) 9, 10, 15, 22, 40, 50, 60, 95, 23, 25, 27, 29

(c) 29, 15, 9, 10, 25, 22, 23, 27, 40, 60, 50, 95

(d) 95, 50, 60, 40, 27, 23, 22, 25, 10, 9, 15, 29

188. How many distinct binary search trees can be created out of 4 distinct keys? **[2005, 2 Marks]**

(a) 5 (b) 14

(c) 24 (d) 42

189. A program takes as input a balanced binary search tree with n leaf nodes and computes the value of a function g (x) for each node x. If the cost of computing g (x) is min (number of leaf-nodes in left – subtree of x,number of leaf – nodes in right – subtree of x) then the worst – case time complexity of the program is **[2004, 2 Marks]**

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

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

190. Consider the label sequences obtained by the following pairs of traversals on a labelled binary tree. Which of these pairs identify a tree uniquely? **[2004, 2 Marks]**

(i) Preorder and postorder

(ii) Inorder and postorder

(iii) Preorder and inorder

(iv) Level order and postorder

(a) (i) only (b) (ii) and (iii)

(c) (iii) only (d) (iv) only

191. The following numbers are inserted into an empty binary search tree in the given order : 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum distance of a leaf node from the root)? **[2004, 1 Mark]**

(a) 2 (b) 3

(c) 4 (d) 6

192. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree? **[2003, 1 Mark]**

(a) 7 5 10 3 2 4 6 8 9 (b) 0 2 4 3 1 6 5 9 8 7

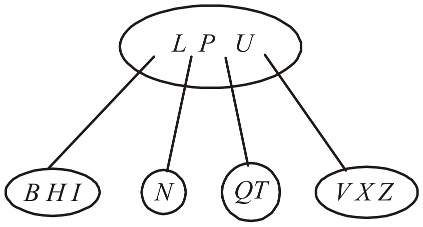
(c) 0 1 2 3 4 5 6 7 8 9 (d) 9 8 6 4 2 3 0 1 5 7

193. Let $T(n)$ be the number of different binary search trees on n distinct elements. Then, $T(n) = \sum_{k\ 1}^{n} T(\text{K}-1)\,T(x)$, where x is **[2003, 1 Mark]**

(a) $n - k + 1$ (b) $n - k$

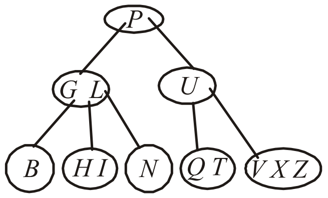
(c) $n - k - 1$ (d) $n - k - 2$

194. Consider the following 2 – 3 – 4 tree (*i. e., B* – tree with a minimum degree of two) in which each data item is a letter. The usual alphabetical ordering of letters is used in constructing the tree. **[2003, 2 Marks]**

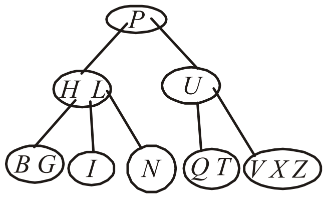


What is the result of inserting G in the above tree?

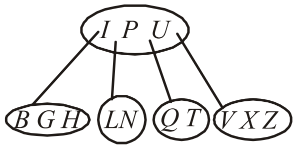
(a)



(b)



(c)



(d) None of these

195. A weight – balanced tree is a binary tree in which for each node, the number of nodes in the left sub tree is at least half and at most twice the number of nodes in the right sub tree. The maximum possible height (number of nodes on the path from the root to the furthest leaf) of such a tree on n nodes is best described by which of the following? **[2002, 2 Marks]**

(a) $\log_2 n$ (b) $\log_{4/3} n$

(c) $\log_3 n$ (d) $\log_{3/2} n$

196. Let LASTPOST, LASTIN and LASTPRE denote the last vertex visited in a postorder, inorder and preorder traversal. Respectively, of a complete binary tree. Which of the following is always true? **[2000, 2 Marks]**

(a) LASTIN = LASTPOST

(b) LASTIN = LASTPRE

(c) LASTPRE = LASTPOST

(d) None of these

197. Consider the following nested representation of binary trees: (XYZ) indicates Y and Z are the left and right sub trees, respectively, of node X. Note that Y and Z may be NULL, or further nested. Which of the following represents a valid binary tree? **[2000, 1 Mark]**

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

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

Binary Heap

198. 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? **[2016, Set 1, 2 Mark]**

(a) $O(1)$

(b) $O(d)$ but not $O(1)$

(c) $O(2^d)$ but not $O(d)$

(d) $O(d\,2^d)$ but not $O(2^d)$

199. 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 _______. **[2016, Set 2, 2 Marks]**

200. 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 **[2015, Set 1, 2 Marks]**

(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

201. Consider the following array of elements

(89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100)

The minimum number of interchanges needed to convert it into a max-heap is **[2015, Set 3, 1 Mark]**

(a) 4 (b) 5

(c) 2 (d) 3

202. A priority queue is implemented as a 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: **[2014, Set-2, 1 Mark]**

(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

203. The number of elements that can be sorted in $\Theta(\log n)$ time using heap sort is **[2013, 2 Marks]**

(a) $\Theta(1)$ (b) $\Theta\sqrt{\log n}$

(c) $\Theta\left(\frac{\log n}{\log \log n}\right)$ (d) $\Theta(\log n)$

204. Which languages necessarily need heap allocation in the runtime environment? **[2011, 2 Marks]**

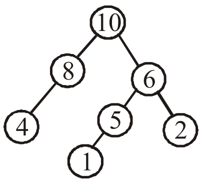
(a) Those that support recursion

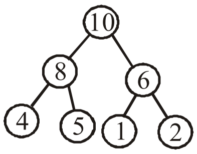
(b) Those that use dynamic scoping

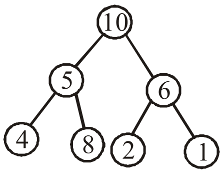
(c) Those that allow dynamic data structures

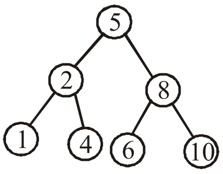
(d) Those that use global variables

205. 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? **[2011, 1 Mark]**

(a)


(b)


(c)


(d)


Statements for Linked Answer Questions 206, 207 and 208:

Consider a binary max-heap implemented using an array:

206. Which one of the following array represents a binary max-heap? **[2009, 1 Mark]**

(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}

207. What is the content of the array after two delete operations on the correct answer to the previous question? **[2009, 1 Mark]**

(a) {14, 13, 12, 10, 8} (b) {14, 12, 13, 8, 10}

(c) {14, 13, 8, 12, 10} (d) {14, 13, 12, 8, 10}

208. 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? **[2009, 1 Mark]**

(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	
2	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	

209. 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 **[2008, 2 Marks]**

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

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

210. In a binary max heap containing n numbers, the smallest element can be found in time **[2007, 2 Marks]**

(a) O (n) (b) O (log n)
(c) O (log log n) (d) O (1)

211. Consider the process of inserting an element into a Max Heap, where the Max Heap is represented by an array. Suppose we perform a binary search on the path from the new leaf to the root to find the position for the newly inserted element, the number of comparisons performed is **[2007, 2 Marks]**

(a) $\Theta(\log_2 n)$ (b) $\Theta(\log_2 \log_2 n)$
(c) $\Theta(n)$ (d) $\Theta(n \log_2 n)$

Statement for Linked Answer Questions 212 and 213

A 3-ary max heap is like a binary max heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows : The root is stored in the first location, a[0], nodes in the next level, from left to right is stored from a[1] to a [3]. The nodes from the second level of the tree from left to right are stored from a[4] location onward. An item x can be inserted into a 3-ary heap containing n items by placing x in the location a[n] and pushing it up the tree to satisfy the heap property.

212. Which one of the following is a valid sequence of elements in an array representing 3 – ary max heap? **[2006, 2 Marks]**

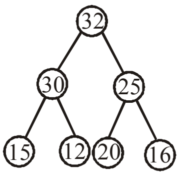
(a) 1, 3, 5, 6, 8, 9
(b) 9, 6, 3, 1, 8, 5
(c) 9, 3, 6, 8, 5, 1
(d) 9, 5, 6, 8, 3, 1

213. Suppose the elements 7, 2, 10 and 4 are inserted in that order, into the valid 3 – ary max heap found in the above question 60. Which one of the following is the sequence of items in the array representing the resultant heap? **[2006, 2 Marks]**

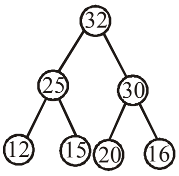
(a) 10, 7, 9, 8, 3, 1, 5, 2, 6, 4
(b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
(c) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3
(d) 10, 8, 6, 9, 7, 2, 3, 4, 1, 5

214. The elements 32, 15, 20, 30, 12, 25, 16, are inserted one by one in the given order into a max-heap. The resultant max-heap is **[2005, 2 Marks]**

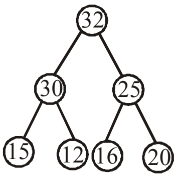
(a)



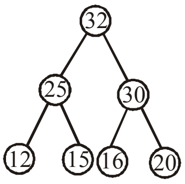
(b)



(c)



(d)



215. A data structure is required for storing a set of integers such that each of the following operations can be done in O(log *n*), time, where *n* is the number of elements in the set

I. Deletion of the smallest element.
II. Insertion of an element, if it is not already present in the set

Which of the following data structures can be used for this purpose? **[2003, 2 Marks]**

(a) A heap can be used but not a balanced binary search tree
(b) A balanced binary search tree can be used but not a heap
(c) Both balanced binary search tree and heap can be used
(d) Neither balanced binary search tree nor heap can be used

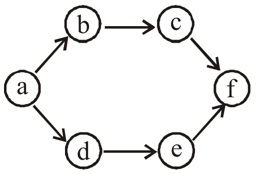
216. Consider an array representation of an *n* element binary heap where the elements are stored from index 1 to index n of the array. For the element stored at index i of the array ($i \le n$), the index of the parent is **[2001, 1 Mark]**

(a) i – 1 (b) $\left\lfloor \frac{i}{2} \right\rfloor$
(c) $\left\lceil \frac{i}{2} \right\rceil$ (d) $\frac{(i+1)}{2}$

Graph

217. *G* is an undirected graph with *n* vertices and 25 edges such that each vertex of *G* has degree at least 3. Then the maximum possible value of *n* is _______. **[2017, Set 2, 1 Mark]**

218. Consider the following directed graph: **[2016, Set 1, 1 Mark]**



The number of different topological orderings of the vertices of the graph is ________ .

219. Let *G* be a weighted connected undirected graph with distinct positive edge weights. If every edge weight is increased by the same value, then which of the following statements is/are **TRUE**? **[2016, Set 1, 1 Mark]**

P: Minimum spanning tree of *G* does not change

Q: Shortest path between any pair of vertices does not change

(a) P only (b) Q only
(c) Neither P nor Q (d) Both P and Q

220. Consider the weighted undirected graph with 4 vertices, where the weight of edge $\{i, j\}$ is given by the entry W_{ij} in the matrix *W*. **[2016, Set 1, 2 Mark]**

$$W = \begin{bmatrix} 0 & 2 & 8 & 5 \\ 2 & 0 & 5 & 8 \\ 8 & 5 & 0 & x \\ 5 & 8 & x & 0 \end{bmatrix}$$

The largest possible integer value of x, for which at least one shortest path between some pair of vertices will contain the edge with weight x is ________.

221. A graph is self-complementary if it is isomorphic to its complement. For all self-complementary graphs on n vertices, n is **[2015, Set 2, 2 Marks]**

(a) A multiple of 4
(b) Even
(c) Odd
(d) Congruent to 0 mod 4, 1 mod 4

222. In a connected graph, a bridge is an edge whose removal disconnects a graph. Which one of the following statements is true? **[2015, Set 2, 2 Marks]**

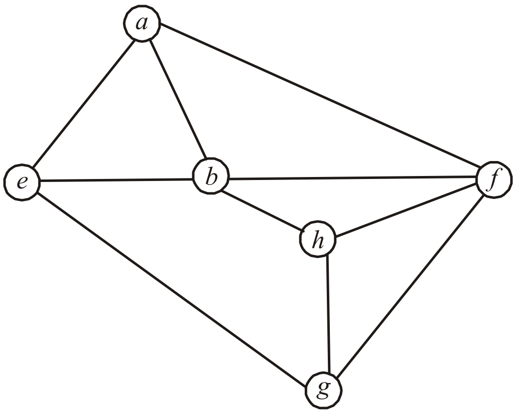
(a) A tree has no bridge
(b) A bridge cannot be part of a simple cycle
(c) Every edge of a clique with size $\geq$ 3 is a bridge (A clique is any complete subgraph of a graph
(d) A graph with bridge cannot have a cycle

223. An ordered n-tuple $(d_1, d_2, .., d_n)$ with $d_1 \geq d_2 \geq ... \geq d_n$ is called *graphic* if there exists a simple undirected graph with n vertices having degrees $d_1, d_2, ..., d_n$ respectively. Which of the following 6-tuples is NOT graphic? **[2014, Set-1, 2 Marks]**

(a) (1, 1, 1, 1, 1, 1) (b) (2, 2, 2, 2, 2, 2)
(c) (3, 3, 3, 1, 0, 0) (d) (3, 2, 1, 1, 1, 0)

224. Consider an undirected graph G where self-loops are not allowed. The vertex set of G is $\{(i, j): 1 \leq i \leq 12, 1 \leq j \leq 12\}$. There is an edge between (a, b) and (c, d) if $|a - c| \leq 1$ and $|b - d| \leq 1$. The number of edges in this graph is_________. **[2014, Set-1, 2 Marks]**

225. Consider the following graph : **[2003, 1 Mark]**



Among the following sequences

1. *abeghf* 2. *abfehg*
3. *abfhge* 4. *afghbe*

Which are depth first traversals of the above graph?

(a) 1, 2 and 4 (b) 1 and 4
(c) 2, 3 and 4 (d) 1, 3 and 4

226. Maximum number of edges in a n-node undirected graph without self loops is **[2002, 1 Mark]**

(a) n^2 (b) $\frac{n(n-1)}{2}$
(c) n – 1 (d) $\frac{(n+1)(n)}{2}$

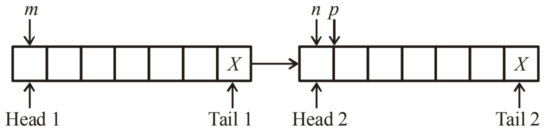
227. How many undirected graphs (not necessarily connected) can be constructed out of a given set $V = \{v_1, v_2, ..., v_n\}$ of n vertices? **[2001, 2 Marks]**

(a) $\frac{n(n-1)}{2}$ (b) 2^n
(c) n! (d) $2^{\frac{n\ n-1}{2}}$

Hints & Solutions

Programming in C

1. (b) According to the given *C* program. It used two linked lists. After the code execution, list '*m*' will be appended to the end of list '*n*' due to pointer '*p*'. Move to last node of the list but in some cases it may dereference to null pointer.



2. (b) Consider all options.
   1. In option (a), p_1 and q_1 are initialized twice or used again for temporary storage, which is not allowed under static single assignment.
   2. In option (b), there is no initialization again of the variables and all the statement are correct.
   3. In option (c), the second line statement is not correct. It should be
   $q_1 = p_1 * \text{C}$.
   And P_2, P_4, q_3 are not initialized anywhere.
   4. In option (d), the second line statements is not correct. It should be
   $q_1 = P_1 * C$
   So, option (b) is correct.

3. (d) Consider all options.
   1. In option (a), we don't need to cast the, result as void * is automatically and safely promoted to any other pointer type in this case. So it is not correct.
   2. In option (b), it is discarded for obvious reason, so it is also not correct.
   3. In option (c), the dangling pointer is nothing but the pointer which is pointing to non-existing memory. (deallocated or deleted memory) which is not happening here, so it is also not correct.
   4. In option (d), when you are calling malloc second time, then new location is assigned to *X* and previous memory location is lost and now we do not have no reference to that location, resulting in memory leak, so option (d) is correct.

4. (c) In the given program, when the function fun 1( ) is calling function fun 2 ( ), after printing value and after returning from fun 2 ( ), it prints the same value.
   In the function fun 2 ( ) also the same thing happens so by looking options we can find the correct sequence of the output after executing $\text{fun}_1(5)$, $(n = 5)$, $\text{fun}_1(n)$, print ⇒ 5.
   $\text{fun}_2\,(n-2) \Rightarrow \text{fun}_2\,(3)$, Print ⇒ 3.
   Similarly all steps execute then, we get the correct.
   Sequence is 5 3 4 2 3 1 2 2 1 3 2 4 3 5
   Hence, option (c) is correct.

5. (c) In the given program, invocation of foo (3) and bar (3), in which, foo (3) turn calls foo (3). This goes on infinite number of times which causes memory overflow and causes abnormal termination.
   bar (3) → bar (2) → bar (1) → bar (0) and (return 0) from here onwards bar (1) will bar (0) and bar (0) will return 0 to bar (1) and this goes on forever without causing memory overflow but it goes on infinite loop execution.
   Hence, option (c) is correct.

6. (3) In the given program.
   X is pointer of string "*abc*" which length is 3.
   Strlen $(S) = 3$
   where (S) is pointer that pointed to *X*.
   Y is pointer of string "*defgh*", which length is 5.
   Strlen $(t) = 5$.
   where *t* is pointer that pointed to *y*, value of *C* is 0.
   Now consider the statement :
   Intlen = [(Strlen (S) – Strlen $(t) > 0)$? Strlen (S) : Strlen (t)]
   $= [(3-5)] > 0$
   $= [-2] > 0$
   It is not true but, since it is given in question
   [Strlen (S) – Strlen (t)] will give unsigned value.
   So, [Strlen (S) – Strlen (t)] $= |-2| = 2$.
   Therefore
   $\Rightarrow 2 > 0$ (it is true) : Strlen (S) : Strlen (t);
   Since $(2 > 0)$ will evaluate true so it will give output as Strlen $(S) = 3$. Hence '3' will be printed.

7. (23) Count in the function Total is static :

i	**Count**	**Total (*i*)**
5	0	2
4	2	3 = (2 + 1)
3	3	5 = (3 + 2)
2	5	6 = (5 + 1)
1	6	7 = (6 + 1)
	Total :	23

Check out the running code with comments
for $(i = 5; i > 0; i--)$
$x = x + \text{total}\ (5)$;
While (5) $(\therefore v = 5)$
Count = Count + *V* & 1
= Count + 5 & 1 (5 & 1 = 1)
Count = Count + 1 (Count = 1)

if $(V = 2)$

Count = 1 + 2 & 1 = 1 + 0 = 1

if $(V = 1)$, count = (1 +(1 & 1)) = 1 + 1

Count = (1 + 1) = 2

Return (2);

Then, $X = 2$ + total (4):

Ccunt = (2 + (4 & 1)) = 2 + 0 = 2

($\therefore$ 2 is static)

If $(V = 2)$

Count = (2 + (2 & 1)) = 2 + 0 = 2

if (v = 1)

Count = (2 + (1 & 1)) = 3

Count = 3

$\boxed{\text{return (3)}}$

X = ((2 + 3) + total (3);)

($\because$ 2 is static)

X = (5 + total (3);)

($\because$ 5 is static)

So, count = (3 + (3 & 1)) = 3 + 1 = 4

If $(V = 2)$

Count = (4 + (2 & 1)) = 4 + 0 = 4

If $(V = 1)$

Count = (4 + (1 & 1)) = 4 + 1 = 5

Count = 5

$\boxed{\text{return (5)}}$

X = (5 + 5) + total (2);

($\because$ 5 is static)

X = (10 + total (2);)

($\because$ 10 is static) so, count = 6 $\boxed{\text{return (6);}}$

X = (10 + 6) + total (1);)

($\because$10 is static) so, count = 7 $\boxed{\text{return (7);}}$

= 16 + total (1);

($\because$ 16 is static), so count = 16 + 7 = **23** (static)

($\because$7 is static)

Hence in the function Total count static is : **23**.

8. (a) 1. Static char var; : Initialization of a variable located in data section of memory because var is defined as character variable whose associated storage class is static.

2. $m = m$ alloc (10); m = Null; A lost memory which cannot be freed because free (m) is missed in code due to 10 contiguous bytes of memory is allocated is address of first byte is stored in 'm' and later it is updated with NULL. Now we lost the address of first bytes of that chunk of memory completely, so we can't free that space as we need the address of first byte to free it.

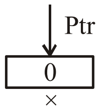
3. Char * Ptr [10]; : Sequence of memory locations to store addresses because ptr is an array of 10 pointers pointing to character variables.

4. Register int var1; : Request to allocate a CPU register to store data because the complier to store the var 1 "value" in CPU Register.

Hence, option (d) is correct.

9. (c) int * Ptr;

X = 0;

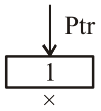


Ptr = & X

Y = *Ptr

$\Rightarrow$(Y = 0)

*Ptr = 1;



$\Rightarrow$(X = 1)

So, the output of invoking print XY(1, 1) is

$\Rightarrow$(1, 0)

Hence, option (c) is correct.

10. (c) The given programe is :

While $(r >= y)$

{

$r = r - y$; (/*statement 1*/)

$q = q + 1$; (/* statement 2*/)

}

Condition given

$X = (Y^*q + r)$

Let q = quotient and r = remainder.

So to divide a number with repeated subtraction, quotient should be initialized to zero and should be incremented for each subtraction.

So, initially $q = 0$, then

$X = (Y^*0 + r)$

$(r = X)$

Since y is subtracted from reach time in given code. In, statement one and q incremented by 1, to avoid undefined behavior. In statement two and the value of y should be greater than zero.

Therefore $(q == 0)$ & & $(r == X)$ & & $(y > 0)$.

Hence, option (c) is correct.

11. (c) In the given program,

First loop will execute 'n' times and the inner loop (j), will execute $\Theta(n \log (n))$ times because for $(i = 1)$, j will run from 1 to n by incrementing by '1' in each step j will run for n times, for ($i = 2$).

Similarly j will run from 1 to n by incrementing by '2', in each step j will run for (n/2) times and so on.

Then the time complexity

$$T_n = \frac{n}{1}+\frac{n}{2}+\frac{n}{3}+\ldots..\frac{n}{(n-1)}\ldots.\log(n-1)$$

$$= n\left[\mathbf{\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\ldots..\frac{1}{(n-1)}}\right]-\log(n-1)$$

$= n$ **log (*n* – 1)** – log (n – 1)

$= n$ log (n – 1)

∴ By neglected negative terms

$= \Theta(n \log n)$

Hence option (c) is correct.

12. (0) Consider each statement in the program

(1) int $m = 10$; //$m = 10$

(2) int n, n_1;

(3) $n = ++m$; //$n = 11$

($n = (++m)$) will increment by m and it assign to n:

($n = 11$, $m = 11$)

(4) $n_1 = m$++; //$n = 11$

$n_1 = m$++; will assign (m) to (n_1) and then increment m by one.

So, ($n_1 = 11$), ($m = 12$)

(5) n– –; //$n = 10$

(n - -), decremented by 1 then ($n = 10$).

(6) - - n_1; //$n_1 = 10$

(- - n_1) decremented by 1 then ($n_1 = 10$)

(7) n - = n_1; //$n = 0$

(n- = n_1), same as $n = (n - n_1) = 10 - 10 = 0$

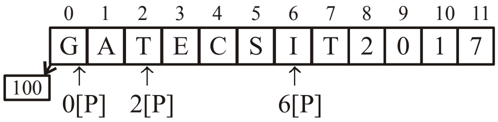
(8) Print$f$("%d", n);

Then, the output will be 0.

Hence, 0 is correct answer.

13. (2) In the given program :

String ⇒ $(C + 2[P] - 6[P] - 1)$



Let, address $((0[P] = 100) = C)$

Strlen $(C + 2[P] + 6[P] - 1)$

= Strlen (100 + ('T' – 'I') – 1)

No, putting ACCII value of T and I, then

= Strlen (100 + 11 – 1)

= Strlen (100 + 10)

= Strlen (110)

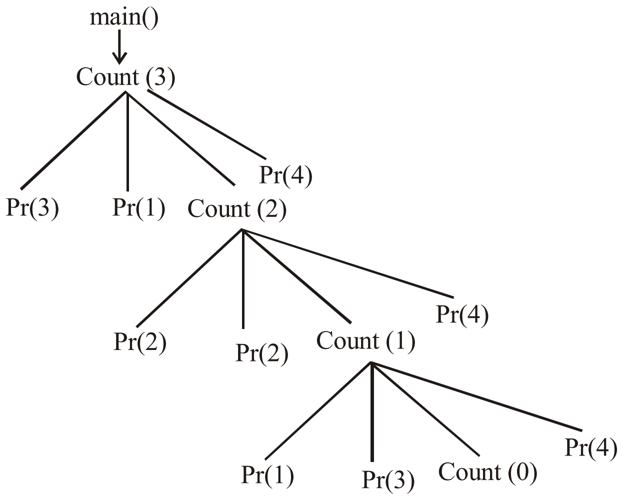
= 2

14. (d)
'i' is integer and *P is value of a pointer to short.

15. 2016
As in C, parameters are passed by value - even if they are pointers. So, here the pointer values are exchanged within the function only (one can use * operator to exchange the values at the location of the pointers and this will affect the values in main). So, there will be no change in a, b, c, d. The output of the program is 2016.

16. (d)
P[a] will have the maximum value of the array, when a = b.

17. (a)



Output will be 312213444.

18. (d)
Dynamic scoping refers to definition of free variable in the reverse order of calling sequence.

19. 9

$2 - 5 + 1 - 7 * 3 \Rightarrow 2 - (5 + 1) - 7 * 3$

$\Rightarrow 2 - 6 - 7 * 3 \Rightarrow 2 - (6 - 7) * 3 \Rightarrow 2 - (-1) * 3$

$\Rightarrow (2 + 1) * 3 \Rightarrow 3 * 3 = 9$

20. 30
The address of i and value of j are passed to the function of f. f modifies i's value to 20. j's value remains same (as its value is passed not the reference). The value printed by the program will be i + j = 20 + 10 = 30.

21. (c) Before Iteration 1: X^Y=64 res * (a^b)=64
Before Iteration 2: X^Y=64 res * (a^b)=64
Before Iteration 3: X^Y=64 res * (a^b)=64
Hence, option (c) is verify.

22. 3 Assume base address of array a is 100.

3	5	2	6	4
100	102	104	106	108

main

⇓

pf (f (100, 5)) 3⇒will be printed

⇓ ⇕ ③

max (f(102, 4), –2)

⇓ ⇕ ③

ret max (f (104, 3), 3)

⇓ ⇕ ②

ret max (f (106, 2), –4)

⇓ ⇕ ②

return max (f (108, 1), 2)

⇓ ⇕ ①

Return 1

23. –5 In function "main ()"
f1 is called by value, so local variables a, b, c of f1 are customized but not the local variables a, b, c of main function.
f2 is called by reference.

```
int main () {
int a = 4, b = 5, c = 6
f1 (a, b)
f2(&b, &c)
printf ("%d", c-a-b);
}
f2(int *a, int *b)
{
int c;
c = * a; c 5
* a = b; [will change 'b' value of main to c value of main]
*b = c; [will change 'c' value of main to c value of f2]
}
```

24. (b) The loop terminater when $r < y$. so, $r < y$ is one post condition.
In each iteration q is incremented by 1 and y is subtracted from r. Initialvalue of r is x.

So, loop iterates $\frac{x}{y}$ times and q will be equal to $\frac{x}{y}$

and r $= x\% y \Rightarrow x = qy + r$
So, (b) is correct answer.

25. (c)

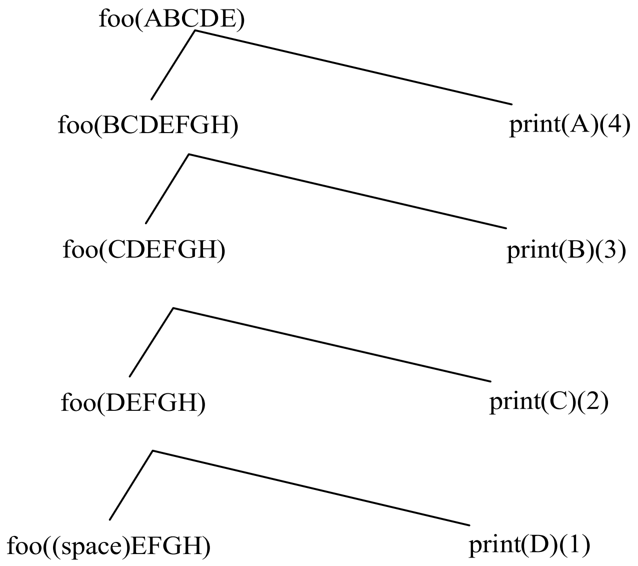
26. (c)

27. 51 Recurrence Relation is
f(n) = 1, if n = 1

$$f(n) = 1 + \sum_{k\ 1}^{n-1} f(k).f(n-k) \text{ if } n > 1$$

n	1	2	3	4	5	6
f(n)	1	2	5	15	51	188

28. (d)

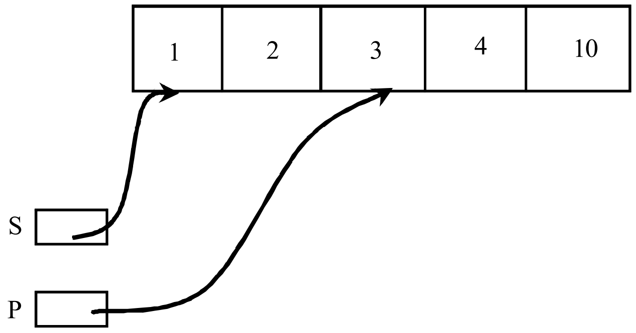


if condition fails & returns controls
∴ DCBA will be pointed

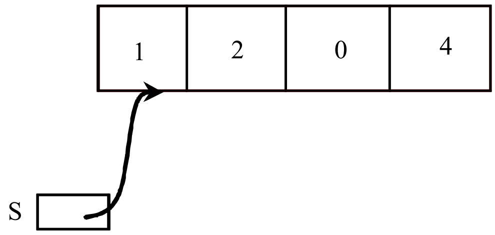
29. 15 The code is pushing 5 and 10 on stack and then popping the top two elements and printing their sum.

30. (a)

31. (c)



After *P = '0', array will become



and we are pointing string S which is 1204.

32. 10 j and k will be evaluated to 2 and – 1 respectively.
In for loop:
When i = 0; 1 time printed (– 1)
When i = 1; 1 time printed (0)
When i = 2; 3 times printed (1, 1, 1)
When i = 3; 3 times printed (2, 2, 2)
When i = 4; 2 times printed (3, 3)
∴ on the whole printf is executed 10 times

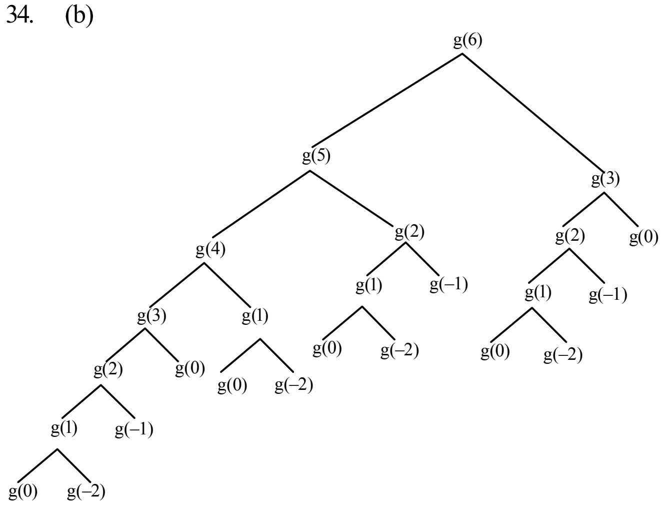
33. 230

In function main x will be updated as follows

$$x = \underset{\substack{\downarrow \\ 1}}{x} + \underset{\substack{\downarrow \\ 26}}{f_1()} + \underset{\substack{\downarrow \\ 51}}{f_2()} + \underset{\substack{\downarrow \\ 100}}{f_3()} + \underset{\substack{\downarrow \\ 52}}{f_2()}x$$

Note: static variable in f_2( ) will be initialized only once & it retains value in between function calls.

34. (b)



Total calls = 25

35. (d) We concentrate on following code segment:

→ int *Pi = & i ;

Nothing wrong as 'Pi' is declared as integer pointer and is assigned the address of 'i' which is an "int".

→ scanf ("%d", Pi);

We know that scanf ( ) has two arguments:

First is control string ("%d" in this case), telling it what to read from the keyboard.

Second is the address where to store the read item. So, 'Pi' refers to the address of "i", so the value scanned by scanf ( ) will be stored at the address of 'i'.

Above statement is equivalent to:

scanf ("%d", & i);

Finally the best statement:

printf ("%d\n", i + 5);

It prints the value 5 more than the value stored in i. So, the program executes successfully and prints the value 5 more than the integer value entered by the user.

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

37. (b) $P = {}^nC_3$

$$= \frac{n(n-1)(n-2)}{6}$$

If we multiply, n, (n – 1) and (n – 2) together, it may go beyond the range of unsigned integer. (∴ option (a) and (d) are wrong)

For all values of n, n(n – 1)/2 will always be an integer value, But for n(n – 1)/3, it is not certain.

Take options (b)

$$P = \frac{n(n-1)/2}{P_1} \times \frac{(n-2)/3}{P_2}$$

P_1 will be having no error, thus P will be more accurate Option (c)

$$P = \frac{n(n-1)/3}{P_1} \times \frac{(n-2)/2}{P_2}$$

There is possibility of truncation in P_1, the accuracy of P is less

38. (d) When j = 50, j (50), the function goes to an infinite loop by calling f(i) recursively for i = j = 50

39. 9 435 ⇒ 110110011

num > > 1

Thus a num is shifted one bit right every time while loop is executed.

While loop is executed 9 times successfully, and 10th time num is zero.

∴ Count is incremented 9 times.

40. 1.72 to 1.74

$$f(g) + q \Rightarrow \frac{x}{2} \quad \frac{1.5}{x} = x$$

$$\Rightarrow \frac{x^2 \quad 3}{2x} = x \Rightarrow x^2 + 3 = 2\,x^2$$

$$\Rightarrow x^2 = 3 \Rightarrow x = 1.73$$

41. 7 P (x) = $x^5 + 4x^3 + 6x + 5$

$= x^3 (x^2 + 4) + 6x + 5$

Now using only one temporary variable 't'

(i) t = x * x (Evaluate x^2 and store in memory)

(ii) t = t + 4 (Evaluate (x^2 + 4) and store in memory)

(iii) t = x^2 (Retrieve x^2 from memory)

(iv) t = t * x (Evaluate x^3 and store in memory)

(v) t = t * (x^2 + 4) (Evaluate x^3 (x^2 + 4) and store in memory)

(vi) t = 6 * x (Evaluate 6x and store in memory)

(vii) t = t + 5 (Evaluate (6x + 5) and store in memory)

(viii) t = t + x^3 (x^2 + 4) (Retrieve x^2 (x^2 + 4) from memory and evaluate {x^3 (x^2 + 4) + 6x + 5}

For the above steps, total number of arithmetic operation is 7

i.e., 4 multiplications and 3 additions.

42. (b) The return value of the function is θ (n^2 log n)

The outer for loop goes $\frac{n}{2}$ + 1 iterations.

The inner for loop runs independent of the outer loop.

And for each inner iteration, $\frac{n}{2}$ gets added to k.

∴ $\frac{n}{2}$ × # outer loops × # inner loops per outer loop.

Inner loops = θ (log n) [∵ $2^{\theta(\log n)}$ = θ (n)]

$$\therefore \frac{n}{2} \times \left[\frac{n}{2} + 1\right] . \theta(\log n) = \theta(n^2 \log n)$$

43. (b) Return value f (p, p) if the value of p is intialized to 5 before the call.

Since, reference of p is passed as 'x'

Thus, any change in value of x in f would be reflected globally.

The recursion can be broken down as

$f(\overset{5}{x},\overset{5}{c})$

$x * f(\overset{6}{x},\overset{4}{c})$

$x * f(\overset{7}{x},\overset{3}{c})$

$x * f(\overset{8}{x},\overset{2}{c})$

$x * f(\overset{9}{x},\overset{1}{c})$

44. (d) Link to activation record of closest lexically enclosing block in program text. It depends on the static program text

45. (c) In switch case statements, there can be more cases, which case satisfied the condition will be executed, so we have to add break statement after every case in switch. If there is no break statement then all switch cases will be executed and default case will also be executed.

In the given program, there is no error so, case A is executed then case B and then case C and then case D and E and finally the default case will be executed and print.

46. (d) * p points to i and q points to j */

```
void f(int *p, int *q)
{
p = q;   /* p also points to j now */
*p = 2;  /* Value of j is changed to 2 now */
}
```

47. (c) 2011 charc[] = "GATE2011";

p now has the base address string "GATE2011"

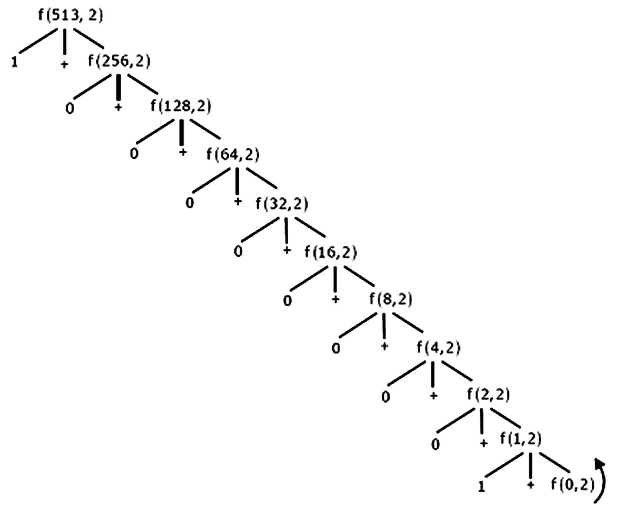
char*p =c;

p[3] is 'E' and p[1] is 'A'.

p[3] - p[1] = ASCII value of 'E' - ASCII value of 'A' = 4

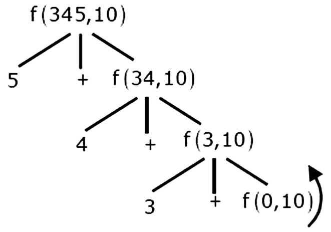
So the expression p + p[3] - p[1] becomes p + 4 which is base address of string "2011

48. (d)



1 + 1 = 2

49. (b)



5 + 4 + 3 = 12

50. (d) In a given program we take the test cases and apply.

First take T_1, if all value equal means

a = b = c = d

So, due to T_1, a = b condition satisfied and S_1 and S_4 executed.

So, from T_2 when all a, b, c, d distinct.

S_1, S_2 not execute, S_3 execute.

from T_3 when a = b then, S_1 execute but c = d so, S_2 not execute but S_3 and S_4 execute but we have no need of T_3 because we get all result from above two.

By using T_4. If a! = b and c = d

So, S_1 not execute and S_2 and S_4 execute so all of S_1, S_2, S_3, S_4 execute and covered by T_1, T_2 and T_4.

51. (b) Let AX, BX, CX be three registers used to store results of temporary variables a, b, c, d, e, f.

a(AX) = 1
b(BX) = 10
c(CX) = 20
d(AX) = a(AX) + b(BX)
e(BX) = c(CX) + d(AX)
f(AX) = c(CX) + e(BX)
b(BX) = c(CX) + e(BX)
e(BX) = b(BX) + f(AX)
d(CX) = 5 + e(BX)
return d(CX) + f(AX)

Thus, only 3 registers can be used without any overwriting of data.

52. (c) f() is a recursive function which adds f(a+1, n–1) to *a if *a is even. If *a is odd then f() subtracts f(a+1, n–1) from *a. See below recursion tree for execution of f(a, 6).

f(add(12), 6) /*Since 12 is first element. a contains address of 12 */

|
|

12 + f(add(7), 5) /* Since 7 is the next element, a+1 contains address of 7 */

|
|

7 – f(add(13), 4)

|
|

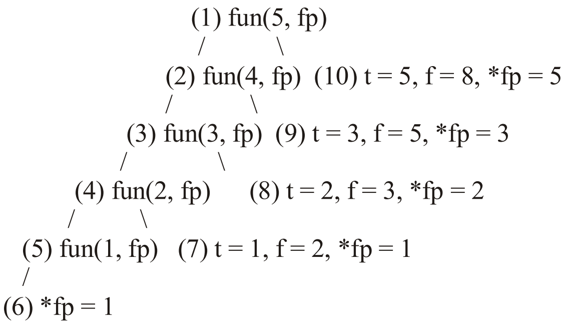
13 – f(add(4), 3)

|
|

4 + f(add(11), 2)
|
|
11 – f(add(6), 1)
|
|
6 + 0

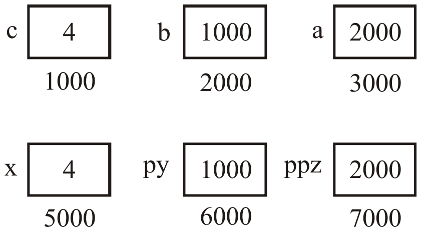
So, the final returned value is
12 + (7 – (13 – (4 + (11 – (6 + 0))))) = 15

53. (b) The program calculates nth Fibonacci Number. The statement t = fun (n-1, fp) gives the (n-1)th Fibonacci number and *fp is used to store the (n-2)th Fibonacci Number. Initial value of *fp (which is 15 in the above program) doesn't matter. Following recursion tree shows all steps from 1 to 10, for exceution of fun(5, &x).


(1) fun(5, fp)
(2) fun(4, fp)
(10) t = 5, f = 8, *fp = 5
(3) fun(3, fp)
(9) t = 3, f = 5, *fp = 3
(4) fun(2, fp)
(8) t = 2, f = 3, *fp = 2
(5) fun(1, fp)
(7) t = 1, f = 2, *fp = 1
(6) *fp = 1

54. (a) The operator "?:" in C is the ternary operator which means that, if the expression is exp 1? exp2 : exp3, so it means, if exp1 is true then exp2 is returned as the answer else exp3 is the required answer.
So, in the given expression let us consider x = 3, y = 4, z = 2,
Then, expression becomes a
= (3 > 4)? ((3 > 2)? 3:2): ((4>2?4:2)
From this, we get that 3>4 is false so we go for the else part of the statement which is 4>2 and is true thus, the answer is 4, the true part of the statement.

55. (b) The program gets executed in the following manner Graphical Representation



Now, considering
int y, z;
**ppy + = 1; z = *ppz = 6
*py + = 2; y = *py = 6
x = 4 + 3 = 7
return x + y + z;
and
c = 4; b & c; a = &b;
printf ("%d", f(c, b, a)),
From the code,
The output is printed as 6 + 6 + 7 = 19.

56. (d) The option chosen is
?1 is ((c = getchar ( ))! = '\n')
?2 is putchar (c);
Because the operator '1=' has higher priority than '=' operator so according to this C = getchar () should be contained in brackets and when the string is reversed then the function putchar (c) is used so that the characters can be printed.

57. (a)
1. Statement is false since global variables are required for recursions with static storage. This is due to unavailability of stack in static storage.
2. This is true
3. In dynamic allocation heap structure is used, so it is false.
4. False since recursion can be implemented.
5. Statement is completely true. So only 2 & 5 are true.

Hence (a) is correct option.

58. (d) From the statement j = j*2 in the code we get to know that j increases in power of 2's. Let us say that this statement execute x times then, according to the question for while loop
2^x < = n
Therefore, x < = $\log_2 n$
And also for termination of while loop there will be an extra comparison required. Thus, total number of comparisons = x + 1
$= \lfloor \log_2 n \rfloor + 2$

59. (d) We follow, the following steps to obtain the value of f(5)

f(5)
r = 5
f(3) + 2 = 18
↑
f(2) + 5 = 16
↑
f(1) + 5 = 11
↑
f(0) + 5 = 6
↑
1

So, f (5) = 1 + 5 + 5 + 5 + 2 = 18

60. (c) S_2: May generate segmentation fault if value at pointer P_x or P_y is constant or P_x or P_y point to a memory location that is invalid. And S_4 : May not work for all inputs as arithmetic overflow can occur.

61. (d) Both functions work1 & work 2 performs the same task, therefore S1 is true.

In S2 it is asking about improvement in performance i.e. reduction in CPU time. When compared work2 will reduce the CPU time, because in work1 a[i+2] is computed twice but in work2 a[i+2] is computed once and stored in t2, and then t2 is used. When we consider the performance in terms of reduction in CPU time, S2 is correct.

62. (b) S_1 : Yes the compiler will generate a temporary nameless cell & initialize it to 13 and pass to swap.

S_2 : No error

S_3 : No error

S_4 : Program will print 13 and 8

S_5 : False.

Hence (b) is correct option.

63. (b) Object Oriented Programming (OPP) is a programming paradigm. The language is object oriented as it use objects. Objects are the data structures that contain data fields and methods together with their interactions.

The main features of the Programming techniques are

1. data abstraction
2. encapsulation
3. modularity
4. polymorphism
5. inheritance

Therefore, the essential features are given by statements (i) and (iv).

64. (c) Languages needs declaration of any statement that we write before its use thus, the common property of both the languages is that both are declarative.

65. (c) An **abstract data type** is a mathematical model for a certain class of data structures that have similar behaviour. An abstract data type is indirectly defined by the operations and mathematical constraints thus, is a user defined data type, not a system defined, that can perform operations defined by it on that data.

66. (c) Syntax to declare pointer to a function => datatype (*pointer_variable)(list of arguments)

To make a pointer to a function => pointer_variable = function_name

Note: don't use parenthesis of the function.

To call (invoke) the function => pointer_variable (list of arguments)

67. (d) P. Functional programming is declarative in nature, involves expression evaluation, & side effect free.

Q. Logic is also declarative but involves theorem proving.

R Object oriented is imperative statement based & have abstract (general) data types.

S. Imperative : The programs are made giving commands & follows definite procedure & sequence.

Hence (d) is correct option.

68. (d) Let us consider below line inside the for loop.

P[i] = S[length – i];

For i = 0, P[i] will be S[6 – 0) and S [6] is ‘10’.

So, P[0] becomes ‘10’. It does not matter what comes in P[1], P[2] .... as P[0] will not change for i > 0.

Nothing is Printed if we print a string with first character ‘10’.

69. (d) When a function is called without being defined, C. Compiler assumes if to reterm “ Int” but here (800) is returning “ double” and hence the compiler will throw type mis-match error. So it is a compiler errors.

70. (a)

71. (c) This is an implementation of Euclid's algorithm to find GCD

Here if m > n, then m = m – n

m < n, then n = n – m

Let take X = 24, Y = 9

Then m = 24, n = 9

iteration	m	n
1	24 – 9 = 15	9
2	15 – 9 = 6	9
3	6	9 – 6 = 3
4	6 – 3 = 3	3

Here m = n, so n returned

Which is GCD (Greatest common divisor) of X&Y

Hence (c) is correct option.

72. (c) Here we take let x = 16

Loop will stop when x – y = 0 or > 0

Iteration	X	Y
1	(16 + 1) / 2 = 8	16 / 8 = 2
2	(8 + 2) / 2 = 5	16 / 5 = 3
3	(5 + 3) / 2 = 4	16 / 4 = 4

Here X=Y

Then take X which is 4.

$(m)^{1/2} = 4 = (16)^{1/2}$

Hence (c) is correct option.

73. (c) The iterations that the given code will undergo are:

From the given conditions we does not take into account when n = 1

Iteration 1

N = 1 + 1 = 2 therefore, i = 2

Iteration 2

N = 2 + 2 = 4 therefore, i = 3

Iteration 3

N = 4 + 3 = 2 = 4 therefore, i = 4

Hence, the value returned after three iterations is 7

When, i = 4 and it also fulfill the condition of n>=5

74. (d) Sum has no use in foo(), it is there just to confuse. Function foo() just prints all digits of a number. In main, there is one more printf statement after foo(), so one more 0 is printed after all digits of n.

From the given code it is found that foo is a recursive function and when the function foo (a, sum) is called where a = 2048 and sum = 0 as per given conditions.

Then, the execution of the function takes in the following manner.

i) k= n % 10 => modulus operator will return the remainder. for example, if n=2048, then 2048 %10 will store the value 8 in k.

ii) j is declared as integer datatype in foo function. Therefore after division if the result contains decimal part, it will be truncated and only the integer part will be stored in j. For example if j=2048/10, (actual result = 204.8) then 204 will be stored in j.

iii) The sum variable declared in the main function will not be used by the foo function.

75. (c) The main goal of structured Programming is to get an understanding about the flow of control in the given program text. In structure programming various control structures such as switch case. If then – else, while, etc.

Allows a programmer to decode the flow of the program easily.

76. (d) Why a, b and c are incorrect? a) call swap (x, y) will not cause any effect on x and y as parameters are passed by value. b) call swap (&x, &y) will no work as function swap() expects values not addresses (or pointers). c) swap (x, y) cannot be used but reason given is not correct. Here the function takes the arguments by value.

Option (a) sends parameter by value but only the local variable a & b will be exchanged but not the actual variables x & y so incorrect.

Option (b) is incorrect sending address of x & y.

Option (c) swap (x, y) is usable there is no need to return.

Option (d) is the opposite statement of option (a), it says that the values are passed by value so won't swap so the option is correct.

77. (b) Balanced parenthesis in an equation are such that the no. of opening and closing parenthesis and in correct order should be there. We can check balancing using stack. When we get any opening parenthesis, then we push that in the stack & if we get a closing one, then we pop the stack. After the complete scanning of input string if stack is found empty then the arithmetic expression is balanced.

78. (b) The function is rewritten as Here initial value of s increments every time with a factor of (p)x/i

Loop Counter (i)	**P**	**S**
1	x	1+x
2	$x * x / 2.1 \quad x^2 / 2.1$	$1 \quad x \quad x^2 / 2$
3	$\frac{x^2}{2} * \frac{x}{3} \quad \frac{x^3}{3}.2.1$	$1 \quad x \quad x^2/2! \quad x^3/3!$
4	$\frac{x^3}{3!} * x/4 \quad x^4/4!$	$1 \quad x \quad x^2/2! \quad x^3/3! \quad x^4/4!$

Thus it can be checked for every value. Here the assumption is that the value of y is very large so y approaches infinity So the series 1+ x + x^2/ 2! + x^3 / 3!...till infinity will have infinite terms & from our previous knowledge we know that this series is expansion of e^x (exponential series) so.

1 + x + x^2/ 2! + x^3/ 3!..........till infinity = e^x

Hence (b) is correct option.

79. (c) (a) True for statically typed languages where each variable has fixed type. Similarly (d) is also correct.

(b) True, in un-typed languages types of values are not defined.

But option (c) is false, since in dynamically typed language variables have dynamically changing types but not that they have no type.

Hence (c) is correct option.

80. (c) The advantages of shared dynamically linked libraries include.

(a) smaller size of executable since less data

(b) lesser overall page fault rate.

(c) No need for re-linking if newer versions of libraries are there.

But since compilation time doesn't include linking so a long linking time required during runtime in DLL ' s so slow startup.

Hence (c) is correct option.

81. (d) 1. Px = newQ();

2. Qy = newQ();

3. Pz = newQ();

4. x : f (1); print 2 # i = 2

5. ((P) y) : f (1);

6. z : f (1) print 2 # i = 2

but line 5. will print 2 because typecast to parent class can't prevent over ridding. So function f(1) of class Q will be called not f (1) of class P.

Hence (d) is correct option.

82. (d) In static scoping the variables are initialized at compile time only

So i = 100 &j = 5

P (i + j) = P (100 + 5) = P(105)

So x = 105

x + 10 = 105 + 10 = 115

So 115 & 105 will be printed.

Hence (d) is correct option.

83. (b) In dynamic scoping, the local values are considered & variables are initialized at run time.

Since x = i + j & in P (x)

i = 200 &j = 20 x = 200 + 20 = 220

& printing (x + 10)

9. = i + j + 10

= 10 + 5 + 10 = 25

Hence (b) is correct option

84. (a) Here X is the global variable so still 5.

Here this is global X whose xy has been changed to 6 so 6 is printed

12 66

First x = 5

Then by function P(&x)

X = 5 +2 = 7

Then by function Q(x)

z = z + x

= 7 + 5 = 12

Here x is global variable so still it is 5.

Return to function P(&x)

Y = 7–1= 6

print x =7

return to main

Print x = 6

Here this is global x whose *y has been changed to 6 so 6 is printed.

85. (d) In call by reference, the called function is working with the memory location of the passed variables. So, any update to the variables are immediately effective.

In call by value-result, the called function is working with a copy of the passed variable. On return, the updated values are copied back to the original variables.

So during a function execution if an exception happens, in call by value-result, the passed variables won't be getting update values.

86. (b) (a) There is no such restriction in C language
(b) True as soon as a function is called its activation record is created in the function stock.
(c) False. In C, variables are statically scoped, not dynamically.
(d) False. The activation records are stored on the same stack.

87. (b) Note the order in which parameters are passed. Inside func1(), x will actually refer to y of main (); and y and z will refer to x of main (). The statement y = y + 4; will result in 14 and statement z = x + y + z will make z = 3 + 14 + 14 = 31 (because y and z point to same variable x of main). Since z refers to x of main(), main will print 31.

88. (c) P1 : Here the function is returning address of the variable x (& x) but the return type is pointer to integer not address. So incorrect.

P2 : *px = 0 directly assigned a value but still px doesn't point to any memory location, so memory initialization or allocation should be done before. So incorrect.

P3: Correction made in P2, memory pre allocated, So correct.

Hence (c) is correct option.

89. (d) n = 10 given but not passed to D. In D, n = 3 &W(n) increments by 1. So n=n+1=4.

Hence (d) is correct option.

90. (c) The process of assigning load addresses to the various parts of the program and adjusting the code and date in the program to reflect the assigned addresses is called relocation. Suppose any default location say x is added to all the addresses in the code leading to correct references. So, it is assembler whose output must distinguish those portions of the instructions and addresses that are relocatable.

91. (d) X - A pointer is assigned to NULL without freeing memory so a clear example of memory leak. Y - Trying to retrieve value after freeing it so dangling pointer. Z - Using uninitialized pointers.

92. (c) X. depth first search is done in stack.
Y. breadth first search is done in queue
Z. sorting is done in a heap.

93. (a) **Aliasing** describes a situation in which a data location in memory can be accessed through different symbolic names in the program. Thus, modifying the data through one name implicitly modifies the values associated to all aliased names, which may not be expected by the programmer. As a result, aliasing makes it particularly difficult to understand, analyze and optimize programs. Thus, multiple variables having same memory location.

94. (c) The amount of memory required to store a structure variable is the sum of the sizes of all its members. But in the case of union, the amount of memory required is the amount required by its largest member. Therefore u, which is a union member of the struct, occupies only 8 bytes of memory, because the largest memory is 8 bytes consumed by long z;. Another member in the struct is short s [5], this will occupy 10 bytes of memory (2 bytes x 5).

95. (c) A token is strictly a string of characters that the compiler replaces with another string of characters. Token are defined as a sort of search and replace. Token can be used for common phrases, page elements, handy shortcuts, etc. Token are defined with either the define or replace tags and can use single or paired tag syntax.

Examples

```
<define%bgcolor%#ccffcc>
    <define (c) & copy;>
<define [email] joe@ cool.com>
    <:replace [mailto] <a hre =
        mailto: [email] > [email] </a>:>
```

96. (a) Count is static variable in incr(). Statement static int count = 0 will assign count to 0 only in first call. Other calls to this function will take the old values of count. Count will become 0 after the call incr(0). Count will become 1 after the call incr(1) Count will become 3 after the call incr(2) Count will become 6 after the call incr(3) Count will become 10 after the call incr(4)

Arrays

97. (5) To find out the smallest index i such that $A[i]$ is 1 by probing the minimum number of location in A. Assume i be the index of first element and j be the index of last element.

By using the binary search algorithm with some changes.

Now,

Find middle element of Array = $A\left[\frac{\text{low+high}}{2}\right]$

Assuming 'A' is an array of 31 elements with '1' and if it is '1', we check the left part recursively and if it is '0', we check the right part of the array recursively.

Which take $\log_2(31)$ comparisons in the worst case, so the total worst case probes is

$\Rightarrow$ Ceil $(\log_2 31) = 5$

($\because$ Ceil means compute fast log base 2 ceiling)

5 probes performed by an optimal algorithm.

98. (3) The initial contents of the Array is :-

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

For first loop : *i* very from (0 to 4), then the contents of the Array are :

i = 0	5	3	1	4	6	2
i = 1	5	3	1	4	6	2
i = 2	5	3	4	1	6	2
i = 3	5	3	4	6	1	2
i = 4	5	3	4	6	2	1

For second loop : *i* very from (5 to 1) then the contents of the Array are :

i = 5	5	3	4	6	2	1
i = 4	5	3	4	6	2	1
i = 3	5	3	6	4	2	1
i = 2	5	6	3	4	2	1
i = 1	6	5	3	4	2	1

Now, since done = 0, then the for loops will execute again.

For first loop : again *i* is very from 0 to 4, then the contents of the Array are :

i = 0	6	5	3	4	2	1
i = 1	6	5	3	4	2	1
i = 2	6	5	4	3	2	1
i = 3	6	5	4	3	2	1
i = 4	6	5	4	3	2	1

For second loop : Again *i* is very from (5 to 1) then the contents of the array are :

i = 5	6	5	4	3	2	1
i = 4	6	5	4	3	2	1
i = 3	6	5	4	3	2	1
i = 2	6	5	4	3	2	1
i = 1	6	5	4	3	2	1

The value of done is still "zero". Hence the for loop will execute again.

First for loop :

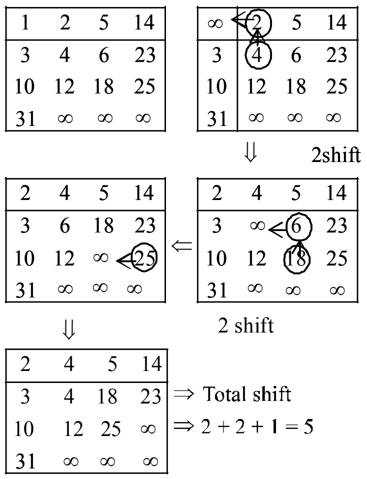
This time there will be no change by the for loop.

Then the value of done is '1', hence the loop terminate here, then the final contents of the Array are :

6	5	4	3	2	1

Hence, the output of the program array [3] is '3'. Hence answer is **3.**

99. 5



1	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

∞	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

2shift

2	4	5	14
3	∞	6	23
10	12	18	25
31	∞	∞	∞

2	4	5	14
3	6	18	23
10	12	∞	25
31	∞	∞	∞

2 shift

2	4	5	14
3	4	18	23
10	12	25	∞
31	∞	∞	∞

⇒ Total shift
⇒ 2 + 2 + 1 = 5

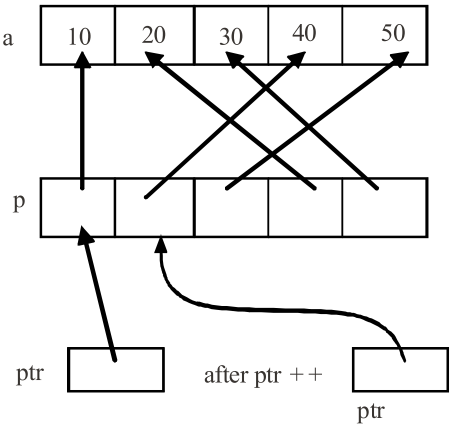
100. 0 C | i | 0 | 1 | 1 |

20 something

{z = 1
for i = 0 + 0.3
do
	k = 0
	z = 1			

z ← z^2 mod	z = 1	z = 4	z = 0	z = 0
if c[i] = 1	c [0] = 1	c [1] = 0	c [1] = 1	c [1] = 1
z ← 2 × z mod 8	z = 2		z = 0	z = 0
end				
return 2				
}				

101. 140



⇒ptr – p = 1 (pointer arithmetic)

** ptr = 40

∴ printf ("%d%d", p + r – p, p + r) will print 140

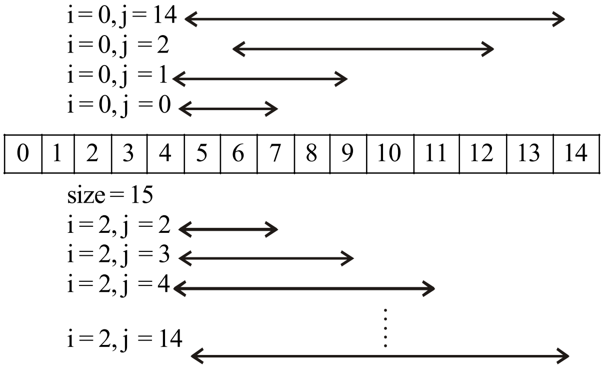
102. (c)

103. (a) Clearly the code segment: for (i = 0; i < size; i + +) Y = Y + E [i]
Computes the sum of all element in the array E.
Note : As the array E may contain negative elements, the sum of elements of subarray of E may be greater than the sum of all elements in the array E.
Now, comes the nested i, j, k loops:
'i' changes the starting of the subway.

'j' changes the end position of the subarray.

'k' loop is used to compute sum of elements of the particular subarray.

The sum of subarray is stored in z and is compared to the earlier max sum Y. Finally the larger sum gets stored in Y.



This can be easily seen how sum of subarray is computed with different values of i and j.

104. (a) For the given pseudo code, each element of the upper triangle is interchanged by the corresponding element in lower triangle and later the lower triangular element is interchanged by the upper triangular once. Thus the final output matrix will be same as that of input matrix A.

105. (c) Analyzing the flow in the function. Fix an 'i' in the outer loop.

Suppose A[i] = old c[0], then A[i] is set to new c[0] [j = 0].

Now suppose that old c[1]= new c[0]

⇒ A[i] = old c[1] (after the j = 0th iteration)

which means in the next iteration with j = 1; A[i] again gets changed form new c[0] (or old c[1]) to new c[1] which is incorrect.

The current loop should "break" after a match is found and A[i] is replaced.

The inner loop should look like

```
for   (int j = 0; j < 3, j + +)
if    (A [i] = = old c[j])
      A[j] = new c [j];
break;
```

The above flow is exposed by the test cases 3 & 4 only. For test 1 and test 2, the expected output is produced by the code.

Test 3 Expected A after replacements

= "abcde" → "acdde"

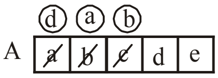
result from code = "abcde" → "addde"

Test 4 Expected "abcde" → "bacde"

result form code "abcde" → "abcde".

∴ Answer is 3 and 4 only (c)

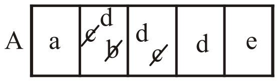
106. (b)



old C	a	b	c
New C	d	a	b

For (1)

Here, when the element of array A and old C match, we replace that array element of A with array element of new C for every element of A array update occurs maximum one time. Similarly for (2) array element of A has updated with array element of new C less than or equal to one time.



old C	b	c	a
New C	c	d	a

For (3)

Now, for (3), when i = O, value of A which old C[2] i.e. 'a' and replace with new C[2] i.e. also 'a', So, no changes.

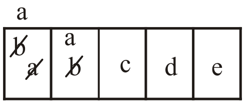
when i = 1 value of array. A[1] = 'b'

Match with Old C [O] = 'b' and replace with new C [O] = 'C'.

Now, A [1] = 'C', which equal with 'd'. Next element of old C [1] = 'C'.

So, replace again with new C [1] = 'd'.

Now, we can say here in Array A [1] value replace with new C [O] value, and that new C [O] value replace with next new C [1] value.



Old C	a	b	c
New C	b	a	c

Similarly for (4) here 2 times replacement for A[O] with element new C [O] and new C [1]. Updating of new C value with another new C value is calling flow here, Hence the option (c) is correct.

107. (b) Dynamic programming can be successfully used, i.e n rows for n elements &w + 1 columns.

Each row is filled on the basis of its previous rows & the $(j - a_i)$ – th column.

If any of them is 0 then X[i, j] should be zero.

This require $X[i, j] = X[i-1, j] \vee \times [i-1, j-a_i]$

Hence (b) is correct option.

108. (c) The algorithm of dynamic programming has n rows & w columns.These would be filled dynamically depending upon previous rows &columns. So X[n,w] will be filled in the last & this would give the result. If X[n,w] = 1 i.e. TRUE, then we know that there is subset present whose sum = integer w.

Otherwise subset not present.

Hence (c) is correct option.

109. (c) We are given that

$i = 0; j = 9;$

$k = (i + j)/2$

I iteration

Let consider $x = 4$

Now, $k = (0 + 9)/2 = 4$

Therefore, $y[4] < x$ is true

→ $i = 4$ and $j = 9$

II iteration

As $i = 4$

Now, $k = (4 + 9)/2 = 6$

Therefore, $y[6] < x$ is true.

→ $i = 6$ and $j = 9$

III iteration

As $i = 6$

Now, $k = (6 + 9) / 2 = 7$

Therefore, $y[7] < x$ is true

→ $i = 7$ and $j = 9$

IV iteration
As $i = 7$
Now, $k = (7 + 9)/2 = 8$
Therefore, $y[8] < x$ is true
→ $i = 8$ and $j = 9$
V iteration
As $i = 8$
Now, k = (8 + 9) / 2 = 8
Therefore, y[8] < x is true
→ $i = 8$ and $j = 9$
As we have proved that 4[8]! = x & $i < j$, the program will remain forever in loop.

110. (a) The correction in the program is needed as in the above solution we can see that the program cannot work properly.
Now, when $y[k] < x$
Increase 1 to $k + 1$
Otherwise, decrease j to $k - 1$
This will ensure that in the while condition, element at k position will be checked for equality.

111. (b) One possible way to do this is we select first element as max & also min. Then we compare it with all others. At most this would take 2n comparisons during linear search. But if we use divide & conquer as for merge sort we have.
$T(n)\ 2T(n/2) + 2$ for $n > 2$
$\mathrm{T(n)} = 3\mathrm{n}/2 - 2$
$= 1.5n - 2$

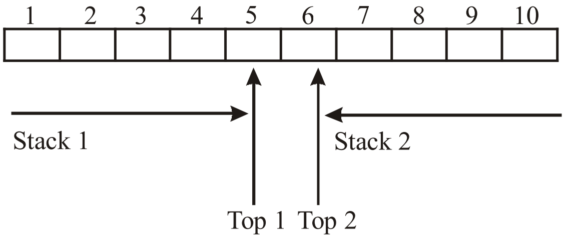
112. (d) The condition given in the FOR loop is
$z[i] = \{x[i] \wedge \sim y[i]\} \vee (\sim x[i] \wedge y[i])$
Now, we clearly can see that the condition reflect the XOR operation and in the XOR operation set obtained is $(X \cap Y') \cup (X' \cap Y)$
Since, the formula is $X \cap Y' = X' - Y$
Result obtained is $(X - Y) \cup (Y - X)$

113. (d) Since the matrices are stored in array, there is no dependence of time complexity on row major or column major. Here only the starting address is known & on the basis of indexes the next memory locations are calculated.
Hence (d) is correct option.

114. (c) The condition (i) is true if the last inserted element in c[] is from a[] and condition (ii) is true if the last inserted element is from b[].

115. (a) We have to store frequencies of scores above 50. That is number of students having score 51, number of students having score 52 and so on. For that an array of size 50 is the best option.

116. (d) Let take MAXSIZE =10



Here the stack will be fuel if both top 1 & top 2 are at the adjacent index values i.e., their difference is 1.
So top 1 = top 2 – 1
Here (d) is correct option.

117. (a) A is an array of pointers to int, and B is a 2D array
* A [2] = Can take a pointer
* A [2] [3] = Can take an Int
* B [1] = It is the base address of array and it cannot be changed, as array in C is a Constant pointer. We can not make B the point some other array. So this is false.
* B[2][3] = It can take an integer.
So, option (a) is correct.

118. (b) If (!A[J]) condition was as follows
(!A[J]) = = 0)
Then, it prints a non zero value in the array
$\{m|m \le n, (Ei)\ [m = i^2]\}$

119. (b) Given is a [40] [50].
In a [40] [50],
Row = 40
Column = 50
The address is 4050.

120. (a) From the given declaration, it is obtained that s is an array denoted by s[10], containing 10 elements which are pointers indicated by the symbol, *, to structure of type node as defined by struct node statement.

121. (a) From the given conditions it can be clearly concluded that, the given sequence rotates s left by k positions.

122. (a) Count = 3

q[1] = 7	q[2] = 3,	q[3] = 9
p[7] = 1	p[3] = 2,	p[9] = 3

(b) The first count elements of q contain values (i) such that set (i) has been called.
(c) Suppose p[i] has a value < = 0 or > count, then to set (i) return false.
Otherwise consider q[p[i]]. Using (b), q[p[i]] contains a value (i) such that set (i) has been called. Since set (i) has not been called, q[p[i]] cannot be equal to (i) and is set (i) returns false.

123. (a) (1) f[n] ! = 0
or
Expression like $f(n)^3$ 0 etc.
(2) f[n]
(3) f[n] = t
(b) O(n) or (n + 1)
Alternately
0 (n^3) in the log-cost model.

124. (a) 47
(b) 22210000 1111
Arrangements that satisfy one of the following two constraints are also correct.
(i) 1's in the first 4 positions and no 2 in the last 3 positions
or

$$\frac{1111}{\text{all ones}}\ 22020\ \frac{010}{\text{no two's}}$$

(ii) 1's in the last 3 positions and no zero in the first 4 positions
or

$$\frac{1212}{\text{no zeros}}\ 00020\ \frac{111}{\text{all ones}}$$

Stacks

125. (b) Viable prefixes appear only at the top of the stack and not inside is valid in shift reduce parsing, because we will write the shift reduce parsing program for every variable such that if any variable contain more than one possibility, it will choose the correct production.

126. (c) The queue can be implemated where ENQUEUE takes a sequence of three instructions (reverse, push, reverse) and DEQUEUE takes a single instruction (pop).

127. (a) The algorithm for evaluating any postfix expression is fairly straightforward:

1. While there are input tokens left

o Read the next token from input.

o If the token is a value

+ Push it onto the stack.

o Otherwise, the token is an operator

(operator here includes both operators, and functions).

* It is known a priori that the operator takes n arguments.

* If there are fewer than n values on the stack

(Error) The user has not input sufficient values in the expression.

* Else, Pop the top n values from the stack.

* Evaluate the operator, with the values as arguments.

* Push the returned results, if any, back onto the stack.

2. If there is only one value in the stack

o That value is the result of the calculation.

3. If there are more values in the stack

o (Error) The user input has too many values.

Let us run the above algorithm for the given expression.

First three tokens are values, so they are simply pushed. After pushing 8, 2 and 3, the stack is as follows 8, 2, 3

When ^ is read, top two are popped and power(2^3) is calculated 8, 8

When / is read, top two are popped and division(8/8) is performed 1

Next two tokens are values, so they are simply pushed. After pushing 2 and 3, the stack is as follows 1, 2, 3

When * comes, top two are popped and multiplication is performed 1, 6

128. (a) The order in which insert and delete operations are performed matters here.

The best case: Insert and delete operations are performed alternatively. In every delete operation, 2 pop and 1 push operations are performed. So, total m+ n push (n push for insert() and m push for delete()) operations and 2m pop operations are performed.

The worst case: First n elements are inserted and then m elements are deleted. In first delete operation, n + 1 pop operations and n push operation are performed. Other than first, in all delete operations, 1 pop operation is performed. So, total m + n pop operations and 2n push operations are performed (n push for insert() and m push for delete())

129. (c) Let us represent stack-life, of ith element as s(i). The ith element will be in stack till (n–i) elements are pushed and popped. Plus one more y for the time interval between the push of ith element and the (i+1)th element. So,

$$\begin{aligned} S(i) &= y + 2(n–i)\ .\ (y + x) \\ &= y + 2(n - i)\ .\ z \\ &= y + 2nZ - 2iz \end{aligned}$$

Where z = (y + x).

Average, stack-file will, $A = \sum\left[\frac{s(i)}{n}\right]$

$$n\,A \quad ny \quad 2\,.\,n\,.\,n\,.\,z - 2\,.\,z\,.\sum i$$

$$n\,A = ny + 2\,.\,n\,.\,n\,.\,z - 2\,.\,z.\left[\frac{n(n+1)}{2}\right]$$

$$n\ A = ny + 2\ .\ n.n.z - z\ (n.n) - n.z$$

$$A \quad = y + 2\ nz - (n + 1)\ .\ z = y + (n–1).\ z$$

$$= y + (n - 1)(y + x) = n(x + y) - x.$$

130. (a) A stack machine implements registers with a stack. The operands of the arithmetic logic unit (ALU) are always the top two registers of the stack and the result from the ALU is stored in the top register of the stack. 'Stack machine' commonly refers to computers which use a Last-in, First-out stack to hold short-lived temporary values while executing individual program statements. The instruction set carries out most ALU actions with postfix (Reverse Polish notation) operations that work only on the expression stack, not on data registers or main memory cells. The same opcode that handles the frequent common case of an add, an indexed load, or a function call will also handle the general case involving complex subexpressions and nested calls

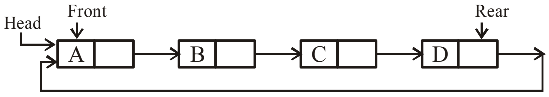
131. (b) 2 stacks . You can even simulate a queue using only one stack. The second (temporary) stack can be simulated by the call stack of recursive calls to the insert method.

The principle stays the same when inserting a new element into the queue:

- You need to transfer elements from one stack to another temporary stack, to reverse their order. Then push the new element to be inserted, onto the temporary stack. Then transfer the elements back to the original stack.
- The new element will be on the bottom of the stack, and the oldest element is on top (first to be popped).

Queues

132. (b) In circular queue next pointer of Rear node pointing to the Front Node will lead to insertion in a queue is always from REAR and deletion is always from FRONT node in Θ(1) time.



Hence, option (B) is correct.

133. (a) We can use circular array to implement both O(1) time.

134. 256

$\because n = 16$

Therefore, maximum number of iterations will be $n^2 = 256$

135. (a) To compute the worst case time complexity of a sequence of n queue operations on an initially empty queue is θ (n).

Complexity of a squence of 'n' queue operations = Total complexity of Enqueue operations (α) + Total complexity of Dequeue operations (β).

Total complexity of Dequeue operations (β) ≤ Total complexity of Enqueue operations

$\beta \le \alpha$ ...(i)

Total complexity of queue operations $(\alpha) \le n$ ...(ii)

Total complexity of n operations = $\alpha + \beta$

$\le \alpha + \alpha$ [From ...(i)]

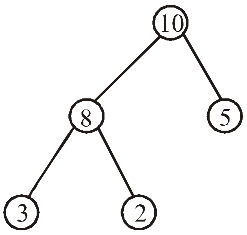
$\le n + n$ [From ... (ii)]

$\le 2\ n$

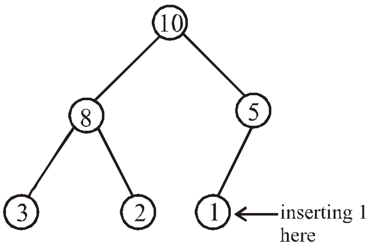
∴ Worst Case Time Complexity of 'n' operations is (a) θ (n).

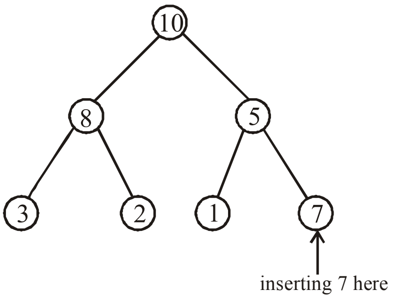
136. (a) FRONT points to the first element that we can remove and then we increment FRONT REAR points to the first empty space in queue so we insert an element and increment REAR so now initially queue is empty F=R=0 insert A (AT 0th index) --> increment REAR (REAR =1 FRONT=0) now if we want to delete A we have FRONT pointing at that location so we delete A and increment FRONT (REAR=1 FRONT=1 queue empty).

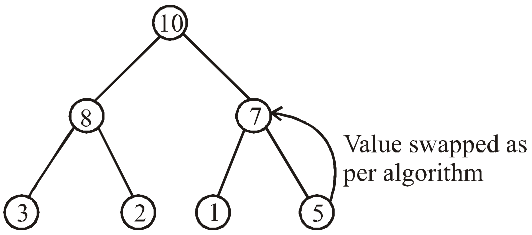
137. (d) Initial level order traversal with 10, 8, 5, 3, 2



Now, let us insert the values







Therefore, the level order traversal comes out to be 10, 8, 7, 3, 2, 1, 5

Linked Lists

138. (c) The time complexity which can put all these operations together will be $O(N^2)$.

139. (d) When the while loop ends, q contains address of second last node and p contains address of last node. So we need to do following things after while loop.

(i) Set next of q as NULL (q->next = NULL).

(ii) Set next of p as head (p->next = head).

(iii) Make head as p (head = p)

Step (ii) must be performed before step (iii). If we change head first, then we lose track of head node in the original linked list.

140. (b) The function rearrange() exchanges data of every node with its next node. It starts exchanging data from the first node itself.

141. (a) Answer is not "(b) front node", as we cannot get rear from front in O(1), but if P is rear we can implement both enqueue and de-Queue in O(1) because from rear we can get front in O(1). Below are sample functions. Shows The pointer points to the rear node:-

Enqueue :- Insert new node after rear, and make rear point to the newly inserted node:

Struct Node * New Node;

New Node → Next = Rear → Next,

Rear → Next = New Node ;

Rear = New Node;

Dequeue :- Delete the front node, and make the second node the front node.

Rear → Next points to the front node.

Front → Next points to the second node.

Struct Node * Front;

Front = Rear → Next;

Rear → Next = Front → Next;

Free (front);

142. (d) Membership & cardinality functions takes constt. time i.e. O(1), but union & intersection require emparison of 1 element with all the other elements so these two would be slowest.

Hence (d) is correct option.

143. (b) Here the return 1 any 1 of the following should be correct.

(a) P == NULL i.e the list is empty (ends)

(b) P → next = NULL i.e., have one element.

(c) P->data <= p->next ->data i.e., the element is smaller than its next element also. This is true for whole list. Since &&f(p "next") is also there.

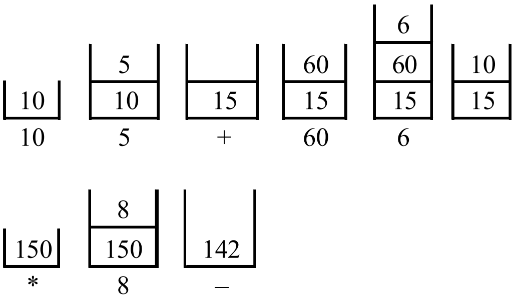
So overall it gives that the elements should be in sorted order.

144. (d) Worst case of searching occurs when the element to be searched is at the end of the list so no. of comparisons required to search complete list would be n. Hence (d) is correct option.

Trees

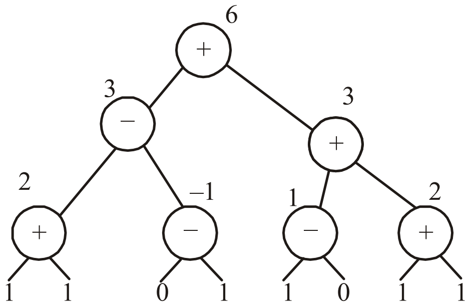
145. (18) A tree with 10 vertices has 9 edges.
Such that, $n = 10, e = (n - 1) = 10 - 1 = 9$
then, $\Sigma \text{degree}(V_i) = 2\,[\text{edges}(E)] = 2 \times 9 = 18$.

146. (c)



147. (a) The in order transversal is as :
left, root, middle, right
∴ Nodes are visited in SQPTRWUV order.

148. 6

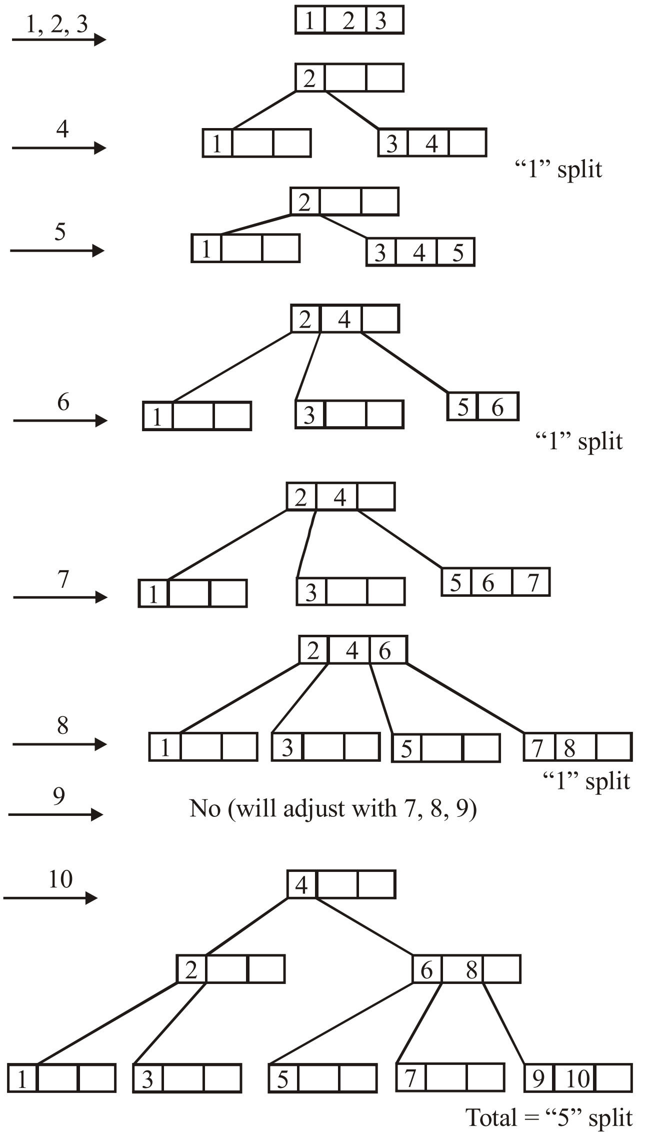


149. (d) R1 $\leftarrow$ c, R2 $\leftarrow$ d, R2 $\leftarrow$ R1 + R2, R1 $\leftarrow$ e, R2 $\leftarrow$ R1-R2
Now to calculate the rest of the expression we must load a and b into the registers but we need the content of R2 later.
So we must use another Register.
R1 $\leftarrow$ a, R3 $\leftarrow$ b, R1 $\leftarrow$ R1-R3, R1 $\leftarrow$ R1+R2

150. (b) AVL trees are binary trees with the following restrictions.
(1) the height difference of the children is at most 1.
(2) both children are AVL trees

```
          a
         / \
        /   \
       b     c
      / \   /
     /   \ /
    d    e  g
   /
  /
 h
```

151. (c) Let's take 10 successive key value {1, 2, 3..........10}



152. (c) Each internal node has n children & so total nodes n
No. of leaf in them is :
$\Rightarrow \; I^* (n - 1)$
$\Rightarrow \; I(n - 1)$
But root can't produce leaf
$I(n - 1) + 1 = L$
$n = L - 1/l + 1$
$n = (41 - 1)/10 + 1 = 5$

153. (c) No. of internal nodes = n
Each node has k children
So total nk
Leaf nodes = nk – n
= n(k – 1)
So considering not node also.
No. of leaf nodes = n(k – 1) + 1
Hence (c) is correct option.

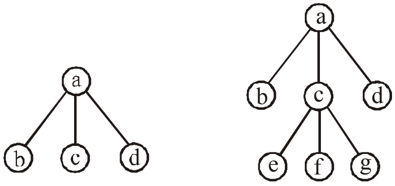
154. (a) a + b × c – d ^ e ^ f
a + b × c – d ^ e f ^
a + b × c – d e f ^ ^
a + b c × – d e f ^ ^
a b c × – d e f ^ ^
a b c × + d e f ^ ^
a b c × + d e f ^ ^–
the result is obtained.

155. (d) Value initialized by 0. If any root node has left child then it adds 1 to the value & move to left child & if any mode has right child also then also calculated the value using recursion & take maximum of both left & right value is taken.

So we know that height is the largest distance between root node & leaf. Therefore, this program calculates heights.

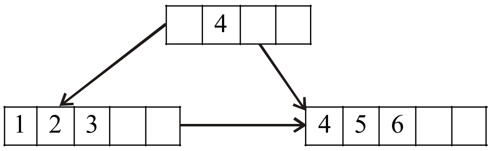
Hence (d) is correct option.

156. (d) Consider following rooted trees

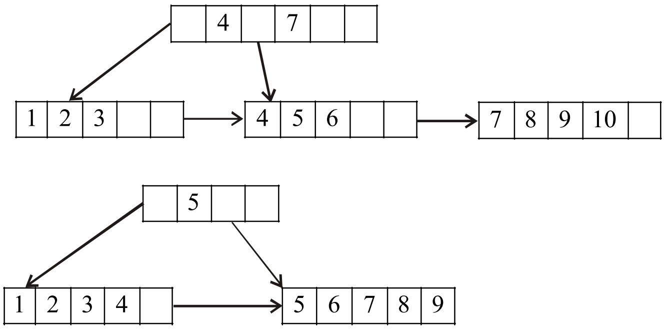


n = 4
(2n + 1)/3 = 3
No. of leaf nodes = 3
Hence (d) is correct option.

157. (a)



(b)

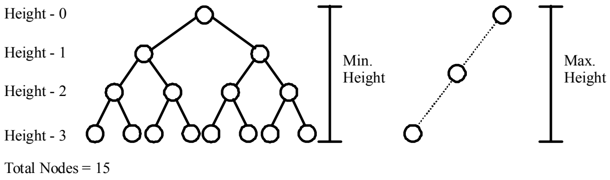


(c) (i) $\left[\frac{m}{2}\right]$ or $\frac{m}{2}$

(ii) m

Binary Tree and Binary Search Tress

158. (b) Since there are 15 nodes, hence the minimum height of the tree will be 3 (when tree will be balanced).



Min. height = floor ($\log_2 N$)
= floor ($\log_2 15$) = 3

The maximum height will be when the tree is skew tree, which will give rise to height 14 either it will be left or right skewed tree. Hence option (b) is correct.

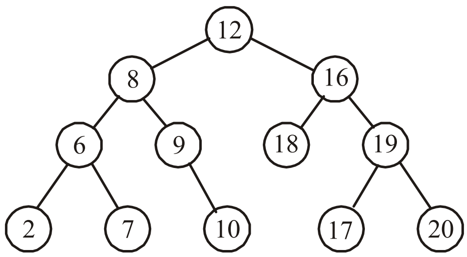
159. (b) For the given pre-order traversal of a binary search tree, Pre order is : (Root, left, Right)

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

In order is : (Left, Root, Right),

So in order of given sequence is : 2, 6, 7, 8, 9, 10, 12, 15, 16, 17, 19, 20.

Then tree will be :



Now, post order is : (Left, Right, Root)

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

Hence option (b) is correct.

160. (52) As given that : Search key = 8 bytes

Block size = 512 bytes.

Block pointer (BP) = 2 bytes.

Then maximum order of B$^+$ tree is

Let K is the order then,

$$K * Bp + (K - 1) * \text{Key} \le \text{Block size}$$

$$K * 2 + (K - 1) * 8 \le 512$$

$$10K \le (512 + 8)$$

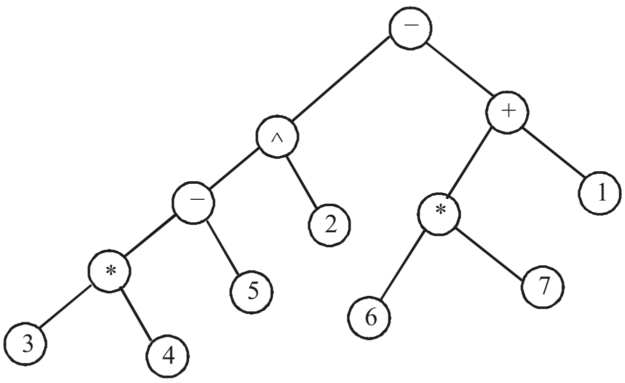
$$10K \le 520$$

$$K = \left(\frac{520}{10}\right) = 52$$

Hence, 52 is correct answer.

161. (a) In B and B+ trees, all the leaf nodes will be at same level.

162. (c) The expression tree for the given post-fix expression is as follows:



New-order of shown expression tree is

$- + 1 * 7\ 6 \wedge 2 - 5 * 4\ 3$

163. 64

To fill 7 levels with 7 elements, at each level we have exactly 2 possible options like 1 and 7 for root one corresponding to making it left skewed and other right skewed. This is valid for all levels upto 6.

Hence, $2^6 = 64$.

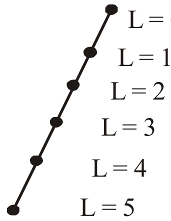
164. (a) We know that the maximum no. of nodes in a binary tree with (height) $h = 2^{h+1}–1$.

Here $h = 5$, then, we easily calculate the h as:

$h = 2^{5+1}–1 = 64 – 1 = 63$

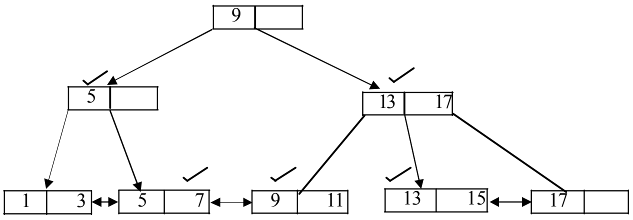
And the minimum no. of nodes with height h is $h + 1$.

$\therefore h = 5$



165. (a) In-order traversal of binary search tree gives ascending orders and in BST, at every node root element is greater than and equal to all element present in left sub-tree and less than or equal to all the elements in right sub-tree.

166. 6



167. (a)

168. 19 Let the number of vertices of a binary tree with ‘p’ leaves be n then the tree has–

(i) p vertices (i.e., leaves) of degree 1

(ii) one vertex (i.e.., root of T) of degree 2

(iii) 'n - p -1' (i.e., interval) vertices of degree 3

(iv) n -1edges

∴ By Handshaking theorem,

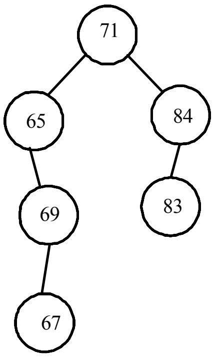
$p \times 1 + 1 \times 2 + (n – p – 1) \times 3 = 2 (n – 1)$

$\Rightarrow$ $n = 2p – 1$

$= 39$ as $p = 20$

$\Rightarrow$ $n – p = 19$ vertices have exactly two children

169. (b)



170. 199

Let the number of leaf nodes of a binary tree with ‘n’ vertices be ‘p’ then the tree has

(i) ‘p’ vertices of degree ‘1’

(ii) one vertex (i.e. root of T) of degree ‘2’.

(iii) 'n – p – 1' vertices of degree ‘3’

(iv) 'n –1' edges

∴ By Handshaking theorem,

$p \times 1 + 1 \times 2 + (n – p –1)\times 3 = 2(n –1)$

$\Rightarrow n = 2\,p – 1$

$= 399$ as $p = 200$

∴ Number of nodes having exactly two children are $n – p$ i.e., 199

171. (c) For non-trivial solution, we have $|A| = 0$

i.e., $\begin{vmatrix} p & q & r \\ q & r & p \\ r & p & q \end{vmatrix} = 0$

$(C_1 \to C_1 + C_2 + C_3)$

$(p + q + r) \begin{vmatrix} 1 & q & r \\ 1 & r & p \\ 1 & p & q \end{vmatrix} = 0$

$[R_2 \to R_2 – R_1; R_3 \to R_3 – R_1]$

$p + q + r = 0$

(or) $\begin{vmatrix} 1 & q & r \\ 0 & r – q & p – r \\ 0 & p – q & q – r \end{vmatrix} = 0$

$\Rightarrow (r – q)^2 – (p – q)\,(p – r) = 0$

$\Rightarrow p^2 + q^2 + r^2 – pq – qr – pr = 0$

$\Rightarrow (p – q)^2 + (q – r)^2 + (r – p)^2 = 0$

$\Rightarrow p – q = 0;\ q – r = 0,\ r – p = 0$

$\Rightarrow p = q = r.$

172. 50

Suppose that ‘k’ is order of the non-leaf node

$k(8)+(k-1)12 \leq 1024$

$20k \leq 1036$

$k \leq \left[\frac{1036}{20}\right] \Rightarrow k \leq 51$

As the order is 51, maximum we can store 50 keys.

173. 1 to 1

Value of a + 10b = 1.

We can find the subtree with 4 nodes in O(n) time, we can use following Algorithm:

(1) Transverse the tree in bottom up manner and find size of subtree rooted with current node.

(2) If size becomes 4, then print the current node.

```
int print 4 subtree (struct Node *root)
{
      if (root = = NULL)
      return 0;
      int l = print 4 subtree (root → left);
      int r = print 4 subtree (root → right);
      if ((l + r + 1) = = 4)
      printf (“%d”, root → data);
      return (l + r + 1)
}
```

174. 110

It takes (log n) time to determine numbers n_1 and n_2 in balanced binary search tree T such that

1. n_1 is the smallest number greater than or equal to L and there is no predecessor n'_1 of n_1 such that n'_1 is equal to n_1.
2. n_2 is the largest number less than or equal to H and there is no successor of n'_2 of n_2 such that is equal to n_2.

Since there are m elements between n_1 and n_2, it takes ‘m’ time to add elements between n_1 and n_2.

So time complexity is O (log n + m)

So the given expression becomes O ($n^0 \log' n + m' \log^0 n$)

And a + 10b + 100c + 1000d = 0 + 10 × 1 + 100 × 1 + 1000 × 1 = 10 + 100 = 110

Because a = 0, b = 1, c = 1 and d = 0

175. (c) The tightest upper bound that represents the time complexity of inserting an object into a binary search tree with n nodes is O (n).

176. (d)

177. (a) The box B1 gets executed when left subtree of n is NULL and right subtree is not NULL. In this case, height of n will be height of right subtree plus one.

The box B2 gets executed when both left and right subtrees of n are not NULL. In this case, height of n will be max of heights of left and right sbtrees of n plus 1.

178. (b) It is stated that there is a binary tree and we have populate the tree with n elements. Sorting the n elements in the increasing order,and placing them in the inorder traversal nodes of the binary tree makes it only BST possible.

179. (a) Such a binary tree is full binary tree (a binary tree where every node has 0 or 2 children).

180. (b) To construct a BST from post order we also require in-order traversal since given the elements are 1 2.......n. So their sorted order would be in order. Using both BST can be constructed in a linear scan. So it will take only 4n time.

181. (d) For a right skewed binary tree, number of nodes will be 2^ (n – 1). For example, in below binary tree, node 'A' will be stored at index 1, 'B' at index 3, 'C' at index 7 and 'D' at index 15.

```
A
 \
  \
   B
    \
     \
      C
       \
        \
         D
```

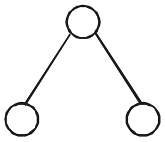
182. (b) It is given that n = 3

$\frac{1}{n+1}\left({}^{2n}C_n\right)$

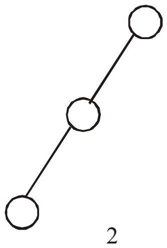
$\frac{1}{3+1}\left({}^{6}C_3\right)$

$\frac{1.6!}{4 \times 3!3!} = 5$

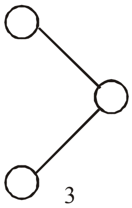
Therefore, there are 5 trees that can be formed with three unlabelled node.



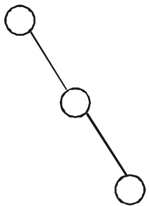
1



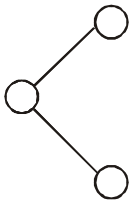
2



3



4



5

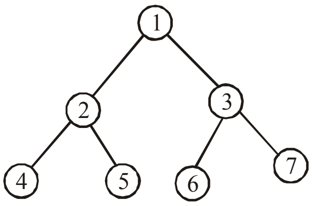
183. (c) This is a formula to calculate the total number of nodes. It is $2^{n+1} - 1$.

Let consider some examples to prove this.

1. Simplest could be taking the binary tree of h (height) = 0. Now, the binary tree of height h will have only 1 node.

Using formula 2 ^ (0 + 1) – 1 = 1. Hence, the formula is correct.

2. Binary tree of h (height) = 2

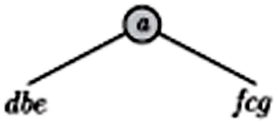


Using formula 2 ^ (2 + 1) – 1 = 7. Hence, the formula is correct.

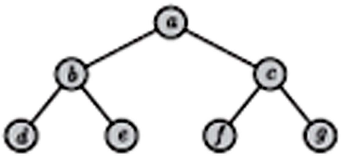
184. (a) In order *d b e a f c g*

preorder *a b d e c f g*

1st element of pre order is root



in preorder *b* is before *d e.* & *c* is before *f g*.



d e b f g c a

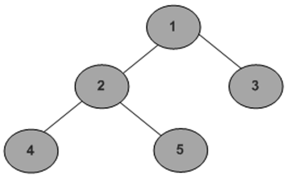
185. (c) A node is a leaf node if both left and right child nodes of it are NULL.

1) If node is NULL then return 0.

2) Else If left and right child nodes are NULL return 1.

3) Else recursively calculate leaf count of the tree using below formula.

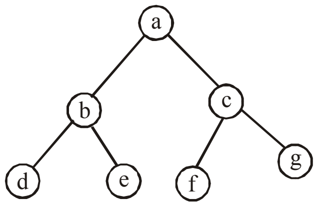
Leaf count of a tree = Leaf count of left subtree + Leaf count of right subtree



Tree

Leaf count for the above tree is 3.

186. (a) The inorder traversal sequence is dbeafcg and the preorder traversal sequence is abdecfg so, the tree is



In the postorder traversal, the sequence is debfgca.

187. (a) Inorder traversal of a BST always gives elements in increasing order. Among all four options, (a) is the only increasing order sequence.

188. (b) The number of keys as per given are 4

Applying the direct formula

$B_n = 1/(n + 1) \times (2n! / n!n!)$

where, B_n is number of binary trees and n is the number of keys.

→ $B_n = 1/(4 + 1) \times (8! / 4!4!)$

→ $B_n = 1/5 \times (8 \times 7 \times 6 \times 5 \times 4!) / 4!4!$

→ $B_n = 8 \times 7 \times 6/(4 \times 3 \times 2)$

→ $B_n = 56/4$

→ $B_n = 14$

The total number of binary trees with $n = 4$ is 14.

189. (b) At the root node (first level) the cost would be $\Theta(n/2)$ as the tree is **balanced.**

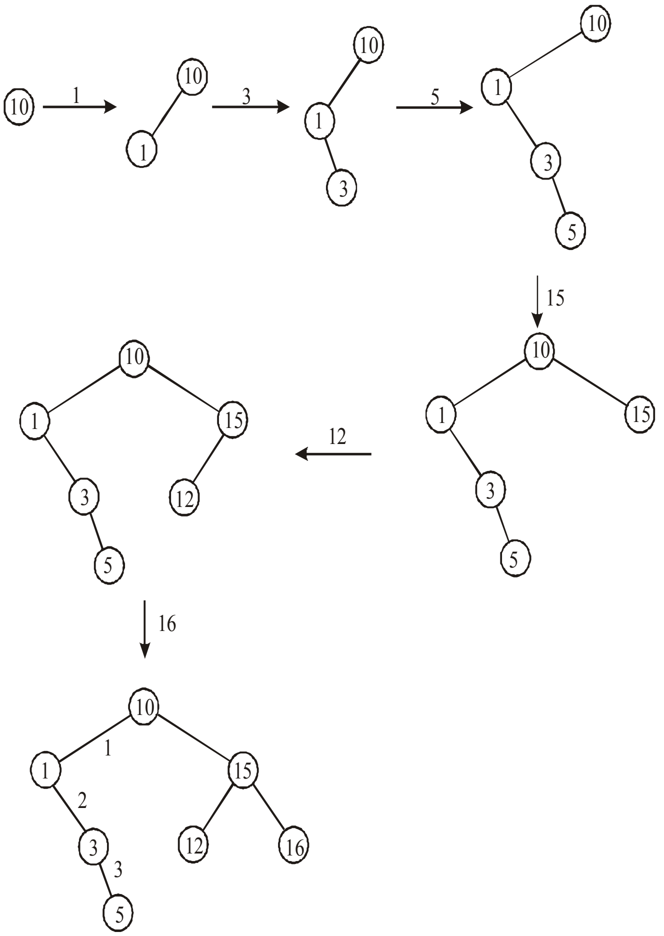
At next level, we have 2 nodes and for each of them cost of computing g(x) will be $\Theta(n/4)$. So, total cost at second level = $\Theta(n/2)$. Similarly at **each level** (total cost per level and not the cost per node in a level) the cost would be $\Theta(n/2)$and so for logn levels it would be $\Theta(n\log n)$.

190. (b) For a tree we not only require in order & preorder but also postorder traversal.

Preorder & postorder help to determine the roots of binary subtrees, inorder arranges those roots in order.

Hence (b) is correct option.

191. (b) Given are 10, 1, 3, 5, 15, 12, 16



The height of the leaf node (5) is high 3.

Hence (b) is correct option.

192. (c) We can solve it in shortcut that the first given element in 7, so we need to choose that particular option in which 7 is at the right place i.e. all the elements on its left should be smaller than it & all the elements on the right should be equal & greater than it.

So this rule is followed in option (c) only.

To make in order of a binary search tree.

(i) Start with the root node.

(ii) Scan its left subtree,

(iii) If the node in subtree has any left child then store the node in stack & repeat this step for its left child unit no. left child of any node.

(iv) If leaf reached then print the node & pop the stack, print the poped value.

(v) Check its right subtree & repeat step (III) for it.

(vi) When stack empty then stop

So here inorder is 0 1 2 3 4 5 6 7 8 9. Actually a fact can be remembered that inorder traversal of a BST leads to a sorted sequence of elements.

Hence (c) is correct option.

193. (b) The summation is for each node, if that node happens to be the root. When a node is root, it will have (k – 1) nodes on the left sub tree (k being any number) and correspondingly (n – k) elements on the right sub tree. So, we can write recurrence T (k – 1) * T(n–k) for the number of distinct binary search trees, as the numbers on left and right sub tree from BSTs independent of each other and only a difference in one of the sub trees produces a difference in the tree. Hence answer is B.

194. (b) Since the given B tree is 2–3–4 tree, there can be at most 4 children or 3 keys. In B tree insertion, we start from root and traverse till the leaf node where key is to be inserted. While traversing. If we find a node which full, we split it. When we inseart G, we find root itself is full. So, we split it when we come down to left most leaf, we find that the leaf is also full, thus we split the leaf also. So, the percent node becomes H. L, P, U and select P as for splitting. Hence option (b) is correct.

195. (d) Let the maximum possible height of a tree with n nodes is represented by H(n).

The maximum possible value of H(n) can be approximately written using following recursion

$H(n) = H(2n/3) + 1$

The solution of above recurrence is log3/2 n. We can simply get it by drawing a recursion tree.

196. (d) It is given that the given tree is complete binary tree. For a complete binary tree, the last visited node will always be same for inorder and preorder traversal. None of the above is true even for a complete binary tree.

The option (a) is incorrect because the last node visited in Inorder traversal is right child and last node visited in Postorder traversal is root.

The option (c) is incorrect because the last node visited in Preorder traversal is right child and last node visited in Postorder traversal is root.

For option (b), see the following counter example.

```
        1
      /   \
     2     3
    / \    /
   4   5  6
```

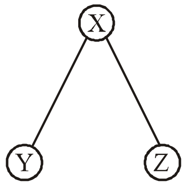
Inorder traversal is 4 2 5 1 6 3

Preorder traversal is 1 2 4 5 3 6

So, lastpost = 1, lastpre = 6, lastin = 3.

Hence option (d) is correct.

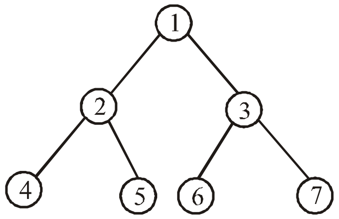
197. (c) (XYZ) indicates that Y is left sub-tree and Z is right subtree. Node is X



As per given in the questions:

(1 (234) (567))

We get, the following tree



1 is the root node

2 and 3 are the non-leaf node

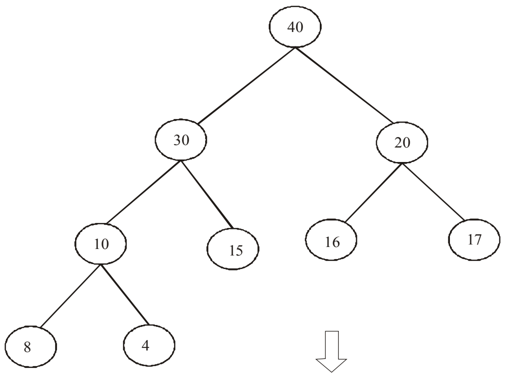
4, 5, 6, 7 are the leaf node which may be null or further nested because in a binary tree every node has 0 or children and not just 1.

Binary Heap

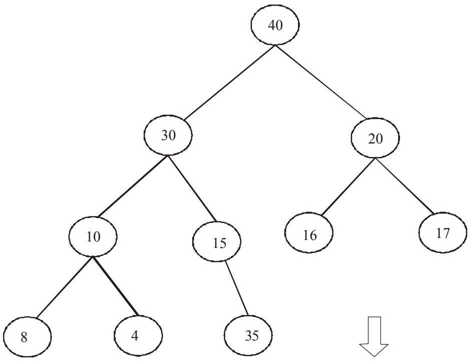
198. (b)

199. 8 Within 'n' levels of min heap, n^{th} smallest element will be present.

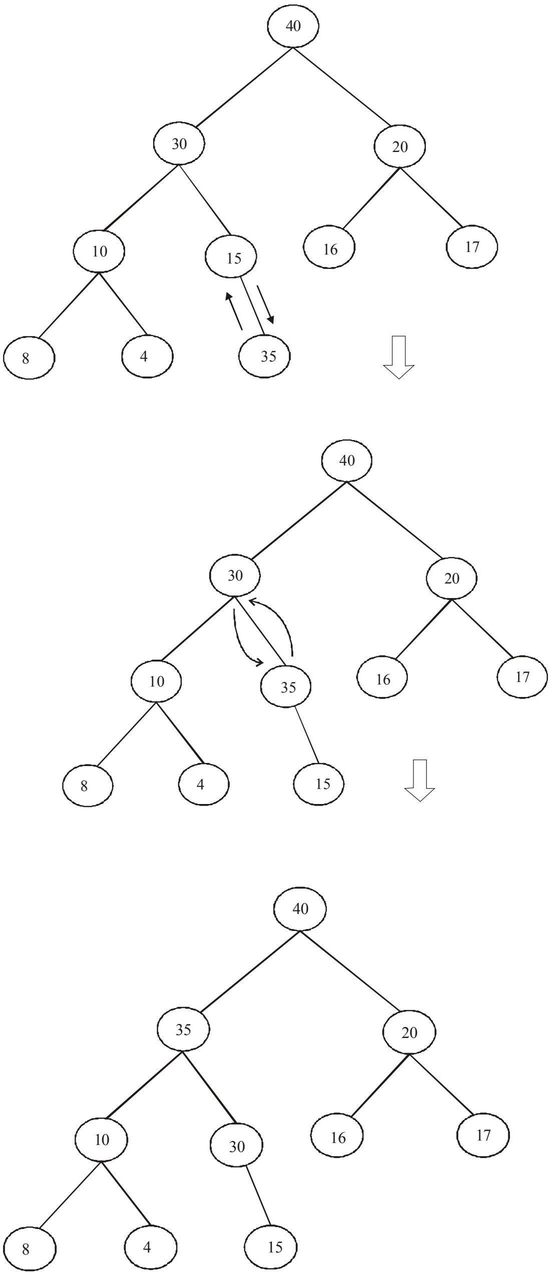
200. (b)



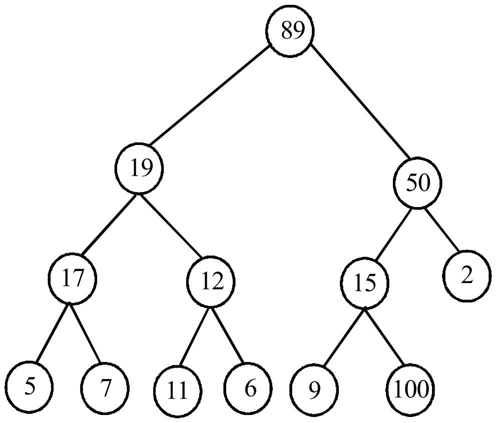
Insert 35 ⇒ according to CBT



Heapification

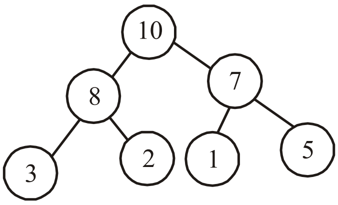


201. **3**



1^{st} swap is : 100 and 15
2^{nd} swap is : 100 and 50
3^{rd} swap is : 100 and 89

202. (a)



After insertion of elements, level - order tansvasal is : 10, 8, 7, 3, 2, 1, 5

203. (c) The number of elements that can be sorted in Θ (log n) time using heap sort is $\Theta\left(\frac{\log n}{\log\log n}\right)$

Consider the number of elements is "k", which can be sorted in θ (k log k) time.

Analyzing the options in decreasing order of complexity since we need a tight bound i.e., θ.

i.e., θ (log n), $\Theta\left(\frac{\log n}{\log\log n}\right), \Theta(\sqrt{\log n}), \Theta(1)$

So if k $\in \Theta$ (log n) time required for heap sort is O (k log k) i.e.,
θ (log n × log log n), But this is not in Θ (log n)

If k $\in \Theta\left(\frac{\log n}{\log\log n}\right)$ time required for Heap Sort

$$\Theta\left(\frac{\log n}{\log\log n} \times \log\left(\frac{\log n}{\log\log n}\right)\right)$$

i.e., $\Theta\left(\log n \times \left[\frac{\log\left(\frac{\log n}{\log\log n}\right)}{\underbrace{\log\log n}_{\leq 1}}\right]\right)$

So, this is in Θ (log n)

Hence, answer is (c) $\Theta\left(\frac{\log n}{\log\log n}\right)$

204. (c) linked **data structures allow** more flexibility in **and the** implementation of **these** linked **data structure** is through **dynamic data structures**

205. (b) A binary tree is max-heap if it is a complete binary tree (A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible) and it follows the max-heap property (value of each parent is greater than or equal to the values of its children).

(a) is not a max-heap because it is not a complete binary tree

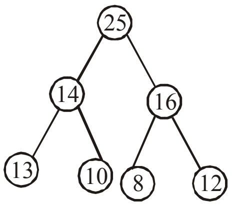
(b) is a max-heap because it is complete binary tree and follows max-heap property.

(c) is not a max-heap because 8 is a child of 5 in this tree, so violates the max-heap property.

(d) is not a max-heap because 8 is a child of 5 in this tree, so violates the max-heap property. There are many other nodes in this tree which violate max-heap property in this tree.

206. (c) Suppose that we have a node x, then for the condition 1 <= x <= n/2 and A[x] >= A[2x + 1] where 2x and 2x + 1 are left child and right child of the node x respectively of a binary max-heap.

Thus, under given cases the tree obtained is.

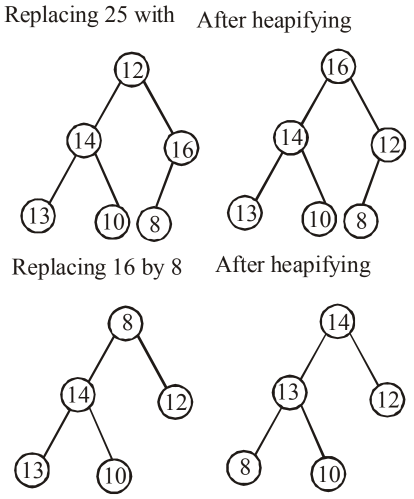


207. (d) Always a greater element is deleted from the binary heap first. The answer of previous question gave the array as [25, 14, 16, 13, 10, 12, 8]

So, when the greatest element, i.e., 25 is deleted the array becomes [14, 16, 13, 10, 12, 8]

And next after second deletion the array becomes [14, 13, 10, 12, 8]

Thus, the procedure for the obtaining the final tree is as follows.


Replacing 25 with

After heapifying

Replacing 16 by 8

After heapifying

208. (c) 12 mod 10 = 2
18 mod 10 = 8
13 mod 10 = 3
2 mod 10 = 2 collision
(2 + 1) mod 10 = 3 again collision
(using linear probing)
(3 + 1) mod 10 = 4
3 mod 10 = 3 collision
(3 + 1) mod 10 = 4 again collision
(using linear probing)
(4 + 1) mod 10 = 5
23 mod 10 = 3 collision
(3 + 1) mod 10 = 4 collision
(4 + 1) mod 10 = 5 again collision
(using linear probing)
(5 + 1) mod 10 = 6
5 mod 10 = 5 collision
(5 + 1) mod 10 = 6 again collision
(6 + 1) mod 10 = 7
15 mod 10 = 5 collision
(5 + 1) mod 10 = 6 collision
(6 + 1) mod 10 = 7 collision
(7 + 1) mod 10 = 8 collision
(8 + 1) mod 10 = 9 collision
So, resulting hash table

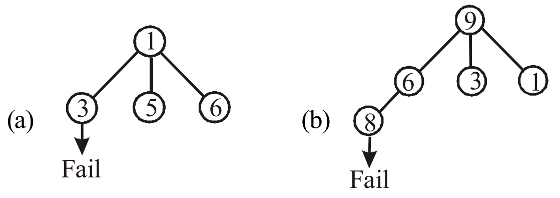
0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

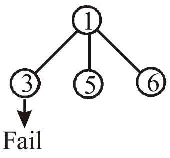
209. (b) Heaps are implemented as simple arrays, to insert *n* more elements each element take $\Theta(1)$ time. So total time would be $\Theta(n)$.

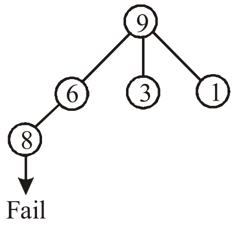
210. (a) Time taken by binary max heap to identify the max element is O(1). Therefore, the time taken by binary max heap to identify the smallest element is O(n).

211. (b) Binary search takes $\Theta(\log2\ n)$, as the search space is continuously divided by two. In inserting in the heap we only have to move up the path from the leaf to the root, the height of a heap cannot be more than $\Theta(\log n)$. So the time complexity for the insertion is $\Theta(\log \log n)$.

212. (d) Create max heap for options.

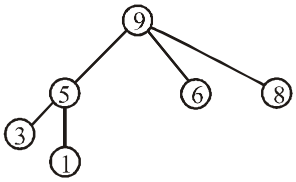


(c) 

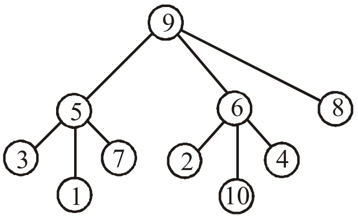
(d) 

Here in options (a), (b) and (c), value present at node is not greater than all its children. Hence (d) is correct option.

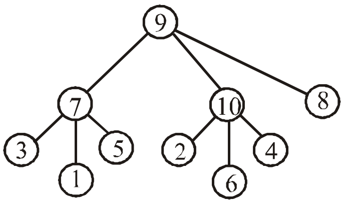
213. (a) Given heap is as follows



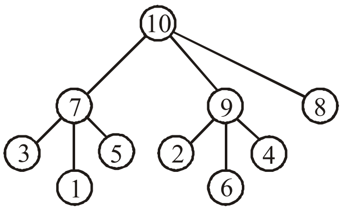
To add 7, 2, 10, 4 we add the node at the end of array



We keep if at right place in the heap tree.
Compare elements with its parent node.
Since 10 > 6 and 7 > 5, we interchange.



Since 10 > 9, we interchange and we get



n / 2 = 10/2 = 5
3 is at right position
4 is at right position
Order
10 7 9 8 3 1 5 2 6 4
Hence (a) is correct option.

214. (a)

215. (b) Both the tasks can be performed by both the data structures but heap is a data structure where to perform these function every element has to be checked so O(n) complexity.

But the balance binary search tree is efficient data structure since at every decision it selects one of its subtree to no. of elements to be checked are reduced by a factor of 1/2 every time.
n /2!= x
x = logn
Hence (b) is correct option.

216. (b) This is a basic question. We know that to reach a node on level *i*, the distance to the root is *i*–1. This implies that if an element is stored at index i of the array, then index of the parent is *n*.

Graph

217. (16) If every vertex has degree at least 3 then, the maximum possible value of *n* where (*n* is the vertices), given edge (*E*) = 25
then, $2(E) \geq \Sigma$ degree of vertices
($\therefore$ According to undirected graph)

$$2(E) \geq 3n$$

$$n \leq \frac{2(E)}{3}$$

$$n \leq \frac{2 \times 25}{3}$$

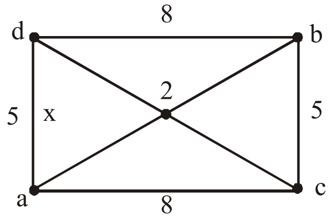
$$n \leq 16.66$$

So, *n* is at Most 16.
Hence 16 is correct answer.

218. 6
The different topological orderings will be as follows:
(1) a b c d e f
(2) a d e b c f
(3) a b d c e f
(4) a d b c e f
(5) a b d e c f
(6) a d b e c f

219. (a) The shortest path may change. The reason is, there may be different number of edges in different paths from s to t. For example, let shortest path be of weight 10 and has 5 edges. Let there be another path with 2 edges and total weight 25. The weight of the shortest path is increased by 5*10 and becomes 10 + 50. Weight of the other path is increased by 2*10 and becomes 25 + 20. So the shortest path changes to the other path with weight as 45.
The Minimum Spanning Tree doesn't change. Remember the Kruskal's algorithm where we sort the edges first. If we increase all weights, then order of edges won't change.

220. 12



The shortest path (excluding x) from 2 to 3 is of weight 12 (2–1–0–3).

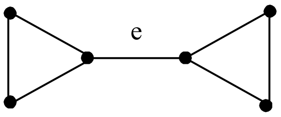
221. (d) An n vertex self complementary graph has exactly half number of edges of the complete graph i.e., $\frac{n(n-1)}{4}$ edges. Since n (n –1) must be divisible by 4, n must be congruent to 0 or 1 module 4.

222. (b) Since, every edge in a tree is bridge

$\therefore$ (a) is false

Since, every edge in a complete graph k_n $(n \geq 3)$ is not a bridge $\Rightarrow$ (c) is false

Let us consider the following graph G:



This graph has a bridge i.e., edge 'e' and a cycle of length '3'

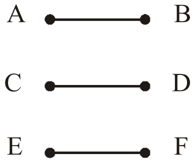
$\therefore$ (d) is false

Since, in a cycle every edge is not a bridge

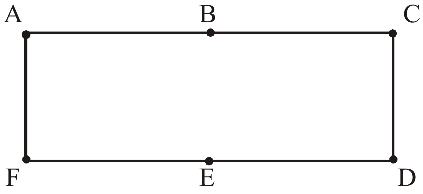
$\therefore$ (b) is true

223. (c) An ordered n type is graphic if it is a degree sequence of some simple undirected graph. options with corresponding graphs are shown below:–

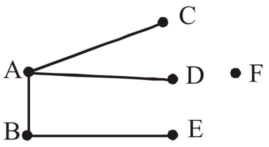
(a) is Graphic seq. (1, 1, 1, 1, 1, 1)



(b) is Graphic seq. (2, 2, 2, 2, 2, 2)



(d) is Graphic seq. (3, 2, 1, 1, 1, 0)



(c) is not graphic seq. (3, 3, 3, 1, 0, 0)

Clearly we can safely remove isolated vertices (i.e. vertices with zero degree) to prepare the graph with this degree sequence. Remaining sequence is (3, 3, 3, 1). Let us name these as A, B, C, D Now vertex with degree 1 (is D)

Must be connect only one A, B or C. If we remove this also then 3 vertices remains A, B and C. One of them will have degree = 2 and other's degree = 3. Now remove vertex with degree 2, finally two vertices remains that must have degree = 2 which is not possible in a simple undirected graph. Hence the seq. (3, 3, 3, 1, 0, 0) is not graphic.

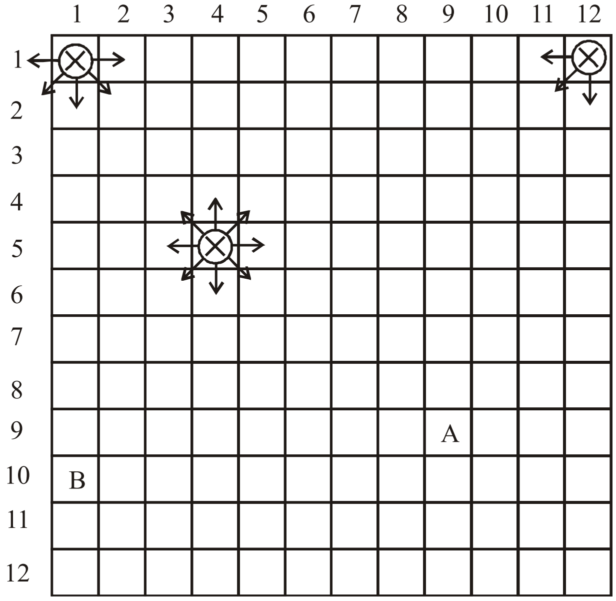
224. 506 edges

The vertices of Graphs are ordered pair (i, j) where two vertices (a, b) and (c, d) are connected by an edge only if $|a - c| \leq 1$ and $|b - d| \leq 1$. Also given that $1 \leq i \leq 12$, $i \leq j \leq 12$.

So, for example (1, 2) is connected to (1, 1), (2, 1), (2, 2), (2, 3) and (1, 3)

Similarly (5, 5) is connected to (4, 4), (4, 5), (4, 6), (5, 4), (5, 6), (6, 4), (6, 5) and (6, 6) and (1, 12) is connected only to (1, 11), (2, 11) and (2, 12).

So, different vertices are connected to different number of vertices. However we can visualize this easily using following diagram:–



Each cell of the table represents corresponding vertex of the graph. For example the cell 'A' represents vertex (9, 0) of the graph, similarly 'B' represents vertex (10, 2) of the graph.

Now, we easily see that there are three kind of vertices in the graph (or table):

(i) Corner vertices which are connected to '3' neighbours.

No. of such vertices = 4

Total no. of edges for such vertices = 4 × 3 = 12

(ii) Vertices in first/ last row of first/last column except corner vertices, which are connected to '5' neighbours each.

No. of such vertices = 40

Total no. of edges for such vertices = 40 × 5 = 200

(iii) Internal cells (vertices) that are connected to '8' neighbours each.

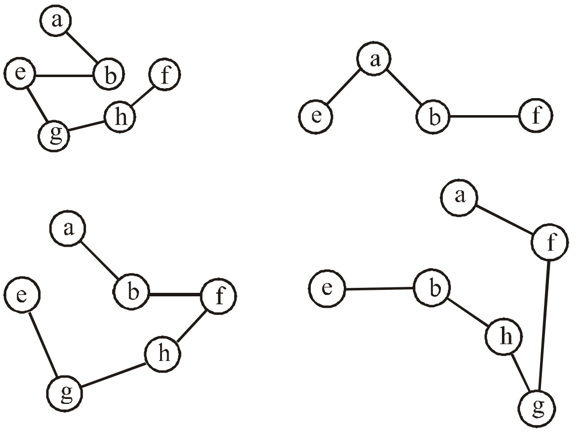
No. of such vertices = 100

Total no. of edges for these vertices = 100 × 8 = 800

Edge total of all above cases = 12 + 200 + 800 = 1012

However in above calculation each edge is counted twice, so, finally total no. of edges in the graph = $\frac{1012}{2} = 506.$

225. (d) DFS traversal takes the path to the end & then move to other branch.



Hence option (d) is correct.

226. (b) Total no. of nodes = n

For an edge of n nodes we select any 2 which make a graph.

So, ${}^{n}c_2 \rightarrow n(n-1)/2$ edges

$n = 4$; $(4 \times 3)/2 = 6$

227. (d) Here, we need to find the number of undirected graphs to be constructed

Now, $S = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 \ldots n - 1$

$= n\,(n-1)/2 \rightarrow$ constructed graphs

$= 2^S$

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