<table>
<tr><td>((MARKS)) (1/2/3...)</td><td>1</td></tr>
<tr><td>((QUESTION))</td><td>

What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    int a[5] = {5, 1, 15, 20, 25};
    int i, j, m;
    i = ++a[1];
    j = a[1]++;
    m = a[i++];
    printf("%d, %d, %d", i, j, m);
    return 0;
}
```

</td></tr>
<tr><td>((OPTION_A))</td><td>2,1,15</td></tr>
<tr><td>((OPTION_B))</td><td>1,2,5</td></tr>
<tr><td>((OPTION_C))</td><td>3,2,15</td></tr>
<tr><td>((OPTION_D))</td><td>2,3,20</td></tr>
<tr><td>((CORRECT_CHOICE))
(A/B/C/D)</td><td>C</td></tr>
<tr><td>((EXPLANATION))
(OPTIONAL)</td><td>

Step 1: int a[5] = {5, 1, 15, 20, 25}; The variable arr is declared as an integer array with a size of 5 and it is initialized to

a[0] = 5, a[1] = 1, a[2] = 15, a[3] = 20, a[4] = 25 .

Step 2: int i, j, m; The variable i,j,m are declared as an integer type.

Step 3: i = ++a[1]; becomes i = ++1; Hence i = 2 and a[1] = 2

Step 4: j = a[1]++; becomes j = 2++; Hence j = 2 and a[1] = 3.

Step 5: m = a[i++]; becomes m = a[2]; Hence m = 15 and i is incremented by 1(i++ means 2++ so i=3)

Step 6: printf("%d, %d, %d", i, j, m); It prints the value of the variables i, j, m

Hence the output of the program is 3, 2, 15

</td></tr>
</table>

((MARKS)) (1/2/3...)	1
((QUESTION))	In C, if you pass an array as an argument to a function, what actually gets passed?
((OPTION_A))	Value of elements in array

((OPTION_B))	First element of the array
((OPTION_C))	Base address of the array
((OPTION_D))	Address of the last element of array
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	The statement 'C' is correct. When we pass an array as a funtion argument, the base address of the array will be passed.

((MARKS)) (1/2/3...)	1

((QUESTION))	Which of the following statements are correct about 6 used in the program?
	int num[6];
	num[6]=21;

((OPTION_A))	In the first statement 6 specifies a particular element, whereas in the second statement it specifies a type.
((OPTION_B))	In the first statement 6 specifies a array size, whereas in the second statement it specifies a particular element of array.
((OPTION_C))	In the first statement 6 specifies a particular element, whereas in the second statement it specifies a array size.
((OPTION_D))	In both the statement 6 specifies array size.
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	The statement 'B' is correct, because int num[6]; specifies the size of array and num[6]=21; designates the particular element(7thelement) of the array.

((MARKS)) (1/2/3...)	1
((QUESTION))	What does the following declaration mean? int (*ptr)[10];
((OPTION_A))	ptr is array of pointers to 10 integers
((OPTION_B))	ptr is a pointer to an array of 10 integers
((OPTION_C))	ptr is an array of 10 integers
((OPTION_D))	ptr is an pointer to array

((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following statements are correct about an array? 1. The array int num[26]; can store 26 elements. 2. The expression num[1] designates the very first element in the array. 3. It is necessary to initialize the array at the time of declaration. 4. The declaration num[SIZE] is allowed if SIZE is a macro.
((OPTION_A))	1
((OPTION_B))	1,4
((OPTION_C))	2,3
((OPTION_D))	2,4
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	1. The array int num[26]; can store 26 elements. This statement is true. 2. The expression num[1] designates the very first element in the array. This statement is false, because it designates the second element of the array. 3. It is necessary to initialize the array at the time of declaration. This statement is false. 4. The declaration num[SIZE] is allowed if SIZE is a macro. This statement is true, because the MACRO just replaces the symbol SIZE with given value. Hence the statements '1' and '4' are correct statements.

((MARKS)) (1/2/3...)	1
((QUESTION))	If the two strings are identical, then strcmp() function returns
((OPTION_A))	-1
((OPTION_B))	1
((OPTION_C))	0
((OPTION_D))	Yes
((CORRECT_CHOICE)) (A/B/C/D)	C

((EXPLANATION)) (OPTIONAL)	strcmp(const char *s1, const char*s2);
	The strcmp return an int value that is
	if s1 < s2 returns a value < 0
	if s1 == s2 returns 0
	if s1 > s2 returns a value > 0

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following function is used to find the first occurrence of a given string in another string?
((OPTION_A))	strchr()
((OPTION_B))	strrchr()
((OPTION_C))	strstr()
((OPTION_D))	strnset()
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	char *strstr(const char *s1, const char *s2); **Return Value**: On success, strstr returns a pointer to the element in s1 where s2 begins (points to s2 in s1). On error (if s2 does not occur in s1), strstr returns null.

	Example:
	#include <stdio.h>
	#include <string.h>
	int main(void)
	{
	   char *str1 = "IndiaBIX", *str2 = "ia", *ptr;
	   ptr = strstr(str1, str2);
	   printf("The substring is: %s\n", ptr);
	   return 0;
	}
	Output: The substring is: iaBIX

((MARKS)) (1/2/3...)	1
((QUESTION))	The library function used to find the last occurrence of a character in a string is
((OPTION_A))	strnstr()
((OPTION_B))	laststr()
((OPTION_C))	strrchr()
((OPTION_D))	strstr()
((CORRECT_CHOICE)) (A/B/C/D)	C

((EXPLANATION)) (OPTIONAL)	**Declaration**: char *strrchr(const char *s, int c);
	It scans a string s in the reverse direction, looking for a specific character c.
	Example:
	#include <string.h>
	#include <stdio.h>
	
	int main(void)
	{
	   char text[] = "I learn through IndiaBIX.com";
	   char *ptr, c = 'i';

	ptr = strrchr(text, c);
	if (ptr)
	printf("The position of '%c' is: %d\n", c, ptr-text);
	else
	printf("The character was not found\n");
	return 0;
	}
	Output:
	The position of 'i' is: 19

((MARKS)) (1/2/3...)	1
((QUESTION))	How will you print \n on the screen?
((OPTION_A))	printf("\n");
((OPTION_B))	echo "\\n";
((OPTION_C))	printf('\n');
((OPTION_D))	printf("\\n");
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	The statement printf("\\n"); prints '\n' on the screen.

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following data structures cannot store non-homogeneous elements?
((OPTION_A))	Arrays
((OPTION_B))	Structure
((OPTION_C))	Linked List

((OPTION_D))	File
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	In arrays ____ and _____ are costly but ____ is easy operation
((OPTION_A))	Searching, insertion, deletion
((OPTION_B))	Insertion, deletion, searching
((OPTION_C))	Deletion, searching, insertion
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	void main()
	{
	int a[5]={1,2};
	printf("\n%d%d%d",a[2],a[3],a[4]);
	}
	What will be the output?

((OPTION_A))	1 2 2

((OPTION_B))	2 1 1
((OPTION_C))	0 0 0
((OPTION_D))	Garbage Value
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	Find the output:
	void main()
	{
	int i=0,a[3];
	a[i]=i++;
	printf(“%d”,a[i]);
	}

((OPTION_A))	0
((OPTION_B))	1
((OPTION_C))	Garbage value
((OPTION_D))	Syntax error
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	While passing the array as actual argument, the function call must have array name_____
((OPTION_A))	alone

((OPTION_B))	With empty braces
((OPTION_C))	With its size
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	Which code will run faster?
	1. for(i=0;i<100;i++)
	    for(j=0;j<10;j++)
	        a[i][j]=0;
	2. for(i=0;i<10;i++)
	    for(j=0;j<100;j++)
	        a[i][j]=0;

((OPTION_A))	Code 1
((OPTION_B))	Code 2
((OPTION_C))	Both run equally
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Consider an integer array int arr[4][5]. If base address is 1020, find the esentation. Size of int is 2 bytes.
((OPTION_A))	1020
((OPTION_B))	1038
((OPTION_C))	1039

((OPTION_D))	1058
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Consider the statement int Val[2][4]={1,2,3,4,5,6,7,8}. The element 4 will be at____
((OPTION_A))	Val[0][3]
((OPTION_B))	Val[0][4]
((OPTION_C))	Val[1][1]
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	void main()
	{
	int a[2][3]={2,3};
	printf(“\n%d%d%d%d”,a[0][0],a[0][1],a[1][0],a[1][1]);
	}
	What will be the output?

((OPTION_A))	0 0 2 3
((OPTION_B))	2 3 0 0
((OPTION_C))	2 0 3 0
((OPTION_D))	2 0 0 3
((CORRECT_CHOICE)) (A/B/C/D)	B

((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The getchar() library function returns________
((OPTION_A))	Character when any key is pressed
((OPTION_B))	Character when enter key is pressed
((OPTION_C))	Display character on the screen when any key is pressed
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	Find the output:
	void main()
	{
	printf("%c",100);
	}

((OPTION_A))	Prints 100
((OPTION_B))	Prints ASCII equivalent of 100
((OPTION_C))	Prints garbage value
((OPTION_D))	Syntax error
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the foolowing is more appropriate for reading a multi-word string?
((OPTION_A))	printf
((OPTION_B))	scanf
((OPTION_C))	put
((OPTION_D))	gets
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	Find the output:
	int main()
	{
	char p[]=”%c\n”;
	p[1]=’d’;
	printf(p,65);
	return 0;
	}

((OPTION_A))	a
((OPTION_B))	A
((OPTION_C))	c
((OPTION_D))	65
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	Sparse matrix have______
((OPTION_A))	Many zero entries
((OPTION_B))	Many non zero entries
((OPTION_C))	High dimension
((OPTION_D))	Many negative entries
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The sequential representation of sparse matrix is given by
((OPTION_A))	stack
((OPTION_B))	Queues
((OPTION_C))	Arrays
((OPTION_D))	Linked List
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	If polynomial is 5x^3+3x^2+10x+2, the degree is
((OPTION_A))	3
((OPTION_B))	2
((OPTION_C))	1
((OPTION_D))	0
((CORRECT_CHOICE)) (A/B/C/D)	A

((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Two main measures for the efficiency of an algorithm are
((OPTION_A))	Processor and memory
((OPTION_B))	Complexity and capacity
((OPTION_C))	Time and space
((OPTION_D))	Data and space
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following case does not exist in complexity theory
((OPTION_A))	Best case
((OPTION_B))	Worst case
((OPTION_C))	Average case
((OPTION_D))	Null case
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The Worst case occur in linear search algorithm when
((OPTION_A))	Item is somewhere in the middle of the array
((OPTION_B))	Item is not in the array at all
((OPTION_C))	Item is the last element in the array
((OPTION_D))	Item is the last element in the array or is not there at all

((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	.

((MARKS)) (1/2/3...)	1
((QUESTION))	The complexity of merge sort algorithm is
((OPTION_A))	O(n)
((OPTION_B))	O(log n)
((OPTION_C))	O(n^2)
((OPTION_D))	O(n log n)
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The input to a merge sort is 6,5,4,3,2,1 and the same input is applied to quick sort then which is the best algorithm in this case
((OPTION_A))	Merge sort
((OPTION_B))	Quick sort
((OPTION_C))	Both have same time complexity in this case as they have same running time
((OPTION_D))	Cannot be decided
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	If there exists two functions f(n) and g(n). The constant c>0 and there exists an integer constant n_0>=1. If f(n)<=c*g(n) for every integer n>= n_0 then we say that____
((OPTION_A))	f(n)=O(g(n))
((OPTION_B))	f(n)=Θ (g(n))
((OPTION_C))	f(n)=Ω (g(n))
f(n)=Θ (g(n))	f(n)=o(g(n))
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	Basic definition of big oh notation

((MARKS)) (1/2/3...)	1
((QUESTION))	In practice ______ is used to define tight upper bound on growth of function f(n)
((OPTION_A))	Big oh
((OPTION_B))	Big omega
((OPTION_C))	Big theta
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	The definition of big oh notation is f(n)<=c*g(n) which defines the upper bound on growth of the function f(n)

((MARKS)) (1/2/3...)	1
((QUESTION))	Examples of O(1) are ______
((OPTION_A))	Multiplying two numbers
((OPTION_B))	Assigning some value to a variable
((OPTION_C))	Displaying some integer on console
((OPTION_D))	All of the above

((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	All these operations are computed by single line expression evaluation

((MARKS)) (1/2/3...)	1
((QUESTION))	Examples of O(n^2) algorithms are
((OPTION_A))	Adding two matrices
((OPTION_B))	Finding transpose of a matrix
((OPTION_C))	Initializing all elements of the matrix by 0
((OPTION_D))	All of the above
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	Within two for loops(nested), all these operations are performed.

((MARKS)) (1/2/3...)	1

((QUESTION))	Choose the correct time complexity of following code__
	while(n>0)
	{
	n=n/10
	}

((OPTION_A))	O(1)
((OPTION_B))	O(n)
((OPTION_C))	O(log n)
((OPTION_D))	O(n^2)
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The time complexity of binary search is_____
((OPTION_A))	O(n)
((OPTION_B))	O(log n)
((OPTION_C))	O(n log n)
((OPTION_D))	O(n^2)
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	The list is divided at the mid and then the element is searched in either left half or right half.

((MARKS)) (1/2/3...)	1
((QUESTION))	Consider recurrence relation as T(0)=c1 T(n)=T(n-1)+c2 This can be expressed as
((OPTION_A))	O(n)
((OPTION_B))	O(log n)
((OPTION_C))	O(n log n)
((OPTION_D))	O(n^2)
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	T(n)=T(n-1)+c2 =T(n-2)+2c2 =T(n-3)+3c2 =T(n-k)+kc2 If k=n then T(n)=c1+nc2 Hence, T(n)=O(n)

((MARKS)) (1/2/3...)	1
((QUESTION))	Consider recurrence relation as T(0)=c1 and T(1)=c2 T(n)=T(n/2)+c3 This can be expressed as
((OPTION_A))	O(n)
((OPTION_B))	O(log n)
((OPTION_C))	O(n log n)
((OPTION_D))	$O(n^2)$
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Following is the method of solving recurrence relation
((OPTION_A))	Greedy method
((OPTION_B))	Backtracking
((OPTION_C))	Forward substitution method
((OPTION_D))	Divide and Conquer method
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The recurrence relation for factorial function is of the form ________
((OPTION_A))	T(n)=T(n-1)+c
((OPTION_B))	T(n)=T(n-1)+T(n-2)+c

((OPTION_C))	T(n/2)+c
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	A

((EXPLANATION)) (OPTIONAL)	The factorial function is as follows-
	fact(n)
	{
	if n=1
	return 1
	else
	return n * fact(n-1)
	}

((MARKS)) (1/2/3...)	1
((QUESTION))	The recurrence relation for fibonacci function is of the form _______
((OPTION_A))	T(n)=T(n-1)+c
((OPTION_B))	T(n)=T(n-1)+T(n-2)+c
((OPTION_C))	T(n/2)+c
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	B

((EXPLANATION)) (OPTIONAL)	The fibonacci function is as follows-
	fibb(n)
	{
	if n = = 0
	return 0
	if n = = 1
	return 1
	else
	return (fibb(n-1) + fibb(n-2))
	}

((MARKS)) (1/2/3...)	1

((QUESTION))	The frequency count of following code is____
	for(i=0;i<m;i++)
	{

	for(j=0;i<n;j++)
	{
	C[i][j]=a[i][j]+b[i][j];
	}
	}

((OPTION_A))	m + mn + mn
((OPTION_B))	m + n + mn
((OPTION_C))	m + n^2 + mn
((OPTION_D))	(m+1) + m(n+1) + mn
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Consider $T(n)=15n^3 + n^2 + 4$. Select the correct statement
((OPTION_A))	$T(n)=O(n^4)$
((OPTION_B))	$T(n)=\Omega (n^3)$
((OPTION_C))	$T(n)=\Omega (n^2)$
((OPTION_D))	All of the above
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1

((QUESTION))	Give the frequency count of 3rd Statement
	for(i=1;i<=n;i++)
	for(j=1;j<=i;j++)
	x=x+1;

((OPTION_A))	½(n^2+n)
((OPTION_B))	½(n^2+3n)
((OPTION_C))	n^2
((OPTION_D))	$(n+1)^2$
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	There are four algorithms for solving a problem. Their time complexities are O(n), O(n2), O(log n) and O(n log n). Which is the best algorithm?
((OPTION_A))	O(n)
((OPTION_B))	O(n^2)
((OPTION_C))	O(log n)
((OPTION_D))	O(n log n)
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The order of the recurrence relation a_r-7a_{r-1}+10a_{r-2}=0 is ______.
((OPTION_A))	3
((OPTION_B))	2
((OPTION_C))	1
((OPTION_D))	B
((CORRECT_CHOICE)) (A/B/C/D)	D

((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Characteristic roots of the recurrence relation a_r-2a_{r-1}+a_{r-2}=0 are _______
((OPTION_A))	1, -1
((OPTION_B))	-1, -1
((OPTION_C))	1, 1
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Charactristic polynomial of the recurrence relation b_n=-3b_{n-1}-b_{n-2} is ________.
((OPTION_A))	Z^2-3Z-2=0
((OPTION_B))	Z^2+3Z-2=0
((OPTION_C))	Z^2+3Z+2=0
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The general solution of the recurrence relation a_r-2a_{r-1}=0 is _____.
((OPTION_A))	a^r=c1$(-2)^r$

((OPTION_B))	$a^r=c2(2)^r$
((OPTION_C))	$a^r=c1(1)^r$
((OPTION_D))	None of these
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Consider the recurrence relation, $a_n=a_{n-1}+2a_{n-2}$ with $a_9=3$ and $a_{10}=5$. Find a_7.
((OPTION_A))	1
((OPTION_B))	3
((OPTION_C))	5
((OPTION_D))	None
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Charactristic polynomial of the recurrence relation $a_{r+2}-a_{r-2}=0$ is ______.
((OPTION_A))	$Z-1=0$
((OPTION_B))	$Z^2-1=0$
((OPTION_C))	$(Z-1)^2=0$
((OPTION_D))	None
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION))	Given homogeneous recurrence relation can be written as

(OPTIONAL)	$a_{r+2}+0a_{r+1}+0a_r+0a_{r-1}-a_{r-2}=0$ Order of this recurrence relation is 4. Hence characteristic polynomial is $Z^4-1=0$

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following is not a homogeneous linear recurrence relation
((OPTION_A))	$a_{r+2}-a_{r-2}=0$
((OPTION_B))	$a_r=a_{r-1}+a_{r-2}$
((OPTION_C))	$a_r-2a_{r-1}=-a_{r-2}$
((OPTION_D))	$a_{r+3}+6a_{r+2}.a_{r+1}-4a_r=0$
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	If 4 and -1 are the characteristic roots of the recurrence relation then its homogeneous solution becomes _______
((OPTION_A))	$a^r=c1(-1)^r+c2(4)^r$
((OPTION_B))	$a^r=c0(-1)^r+c2$
((OPTION_C))	$a^r=(c1+c2.r)(-1)^r$
((OPTION_D))	None
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	A recurrence relation of degree 1 is called ______.
((OPTION_A))	Linear
((OPTION_B))	Homogeneous
((OPTION_C))	Quadratic
((OPTION_D))	None
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The generating function for the sequence 1, a, a^2, a^3, ….. is ______
((OPTION_A))	1/(1-z)
((OPTION_B))	1/(1-az)
((OPTION_C))	1/(1+az)
((OPTION_D))	None
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

MCQs

Suppose we are sorting an array of eight integers using a some quadratic sorting algorithm. After four iterations of the algorithm's main loop, the array elements are ordered as shown here:2 4 5 7 8 1 3 6 *

Insertion sort
Selection sort
either of a and b
none of the above

The running time of insertion sort is *

O(n^2)
O(n)
O(log n)
O(n log n)

Which of the following sorting procedure is the slowest ? *

Quick sort
Heap sort
Shell sort
Bubble sort

A sort which compares adjacent elements in a list and switches where necessary is *

insertion sort
heap sort
quick sort
bubble sort

The correct order of the efficiency of the following sorting algorithms according to their overall running time comparision is *

Insertion>selection>bubble
Insertion>bubble>selection
Selection>bubble>insertion
bubble>selection>insertion

A sort which iteratively passes through a list to exchange the first element with any element less than it and then repeats with a new first element is called *

insertion sort
selection sort
heap sort
quick sort

The number of swappings needed to sort the numbers 8, 22, 7, 9, 31, 19, 5, 13 in ascending order, using bubble sort is *

10
9
13
14

The way a card game player arranges his cards as he picks them one by one can be compared to *

Quick sort
Merge sort
Insertion sort
Bubble sort

Which among the following is the best when the list is already sorted *

Insertion sort
Bubble sort
Merge sort
Selection sort

As part of the maintenance work, you are entrusted with the work of rearranging the library books in a shelf in proper order, at the end of each day. The ideal choice will be *

Bubble sort
Insertion sort
Selection sort
Merge sort