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

((QUESTION))	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;
	}

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

((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(7^{th}element) 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))	The smallest element of an array index called……
((OPTION_A))	Lower bound
((OPTION_B))	Upper bound
((OPTION_C))	Range
((OPTION_D))	None of Above
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

((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))	which of the following is not an asymptotic notation?
((OPTION_A))	BIG-O
((OPTION_B))	Omega
((OPTION_C))	Phi
((OPTION_D))	Theta

((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	
((MARKS)) (1/2/3...)	1
((QUESTION))	Step count for the following loop is For(int i=5; i>0; i++)
((OPTION_A))	5 times
((OPTION_B))	N+1 times
((OPTION_C))	Infinite
((OPTION_D))	No execution
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Time Complexity is
((OPTION_A))	Time required for the machine to compile the program.
((OPTION_B))	Time required for the machine to execute the program.
((OPTION_C))	Time required for the machine to debug the program.
((OPTION_D))	None
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following data structure is not linear data structure?
((OPTION_A))	Array
((OPTION_B))	Linked list
((OPTION_C))	All of above
((OPTION_D))	None of above
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the followings are application of data structure?
((OPTION_A))	Facebook
((OPTION_B))	Searching
((OPTION_C))	Sorting
((OPTION_D))	All of above
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

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

((QUESTION))	Which data structure is used for implementing recursion?
((OPTION_A))	Queue
((OPTION_B))	Stack
((OPTION_C))	Array
((OPTION_D))	List
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Representation of data structure in memory is known as…….
((OPTION_A))	recursive
((OPTION_B))	Abstract Data Type
((OPTION_C))	Storage Structure
((OPTION_D))	File Structure
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	In a linked list with n nodes, the time taken to insert an element after an element pointed by some pointer is……..
((OPTION_A))	O (1)

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

((MARKS)) (1/2/3...)	1
((QUESTION))	In a circular linked list…….
((OPTION_A))	components are all linked together in some sequential manner.
((OPTION_B))	there is no beginning and no end.
((OPTION_C))	components are arranged hierarchically.
((OPTION_D))	forward and backward traversal within the list is permitted.
((CORRECT_CHOICE)) (A/B/C/D)	B
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	Which of the following operators takes only integer operands?
((OPTION_A))	+
((OPTION_B))	/
((OPTION_C))	%

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

((MARKS)) (1/2/3...)	1
((QUESTION))	NULL pointer is used to define .....
((OPTION_A))	End of the linked list
((OPTION_B))	Empty list
((OPTION_C))	Empty pointer field of the structure
((OPTION_D))	All of above
((CORRECT_CHOICE)) (A/B/C/D)	D
((EXPLANATION)) (OPTIONAL)	

((MARKS)) (1/2/3...)	1
((QUESTION))	The function that return memory to heap is called.....
((OPTION_A))	Allloc()
((OPTION_B))	Malloc()
((OPTION_C))	Calloc()
((OPTION_D))	Free()
((CORRECT_CHOICE)) (A/B/C/D)	D

((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)= (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)	

((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)=Loading… (n^3)
((OPTION_C))	T(n)=Loading… (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)) (OPTIONAL)	Given homogeneous recurrence relation can be written as $a_{r+2}+0a_{r+1}+0a_r+0a_{r-1}-a_{r-2}=0$

	Order of this recurrence relation is 4. Hence characteristic polynomial is $Z^4-1=0$

((MARKS)) (1/2/3...)	1
((QUESTION))	The postfix equivalent of the prefix *+ab-cd is _____.
((OPTION_A))	ab+cd-*
((OPTION_B))	ab+cd*-
((OPTION_C))	abcd+*-
((OPTION_D))	ab-cd+*
((CORRECT_CHOICE)) (A/B/C/D)	A
((EXPLANATION)) (OPTIONAL)	

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

((QUESTION))	What does the following function check for? (all necessary headers to be included and function is called from main)
	
	`#define MAX 10`
	
	`typedef struct stack`
	`{`
	`int top;`
	`int item[MAX];`
	`}stack;`
	
	`int function(stack *s)`
	`{`
	`if(s->top == -1)`
	`return 1;`
	`else return 0;`
	`}`

((OPTION_A))	full stack

((OPTION_B))	invalid index
((OPTION_C))	empty stack
((OPTION_D))	infinite stack
((CORRECT_CHOICE)) (A/B/C/D)	C
((EXPLANATION)) (OPTIONAL)	Answer: c Explanation: An empty stack is represented with the top-of-the-stack(‘top’ in this case) to be equal to -1.