VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: D1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST – I Key
WINTER SEMESTER 2025-2026

Programme Name & Branch : **B.Tech & Common to all**
Course Code and Course Name : **BACSE104 - Structured and Object-Oriented Programming**
Faculty Name(s) : **Prof. SENTHIL KUMAR K / Prof. RAMESH BABU VEMULURI / Prof. VIJAYASHREE J /Prof. KEERTHIKA P / Prof. KALAIVANI K / Prof. GOUTAM MAJUMDER / Prof. DHIVYAA C R / Prof. SRI PREETHAA K R / Prof. GUNASEKAR M / Prof. SWETHA V / Prof. SYAMASUDHA VEERAGANDHAM / Prof. ASHOK KUMAR RAI / Prof. JABANJALIN HILDA J / Prof. VINILA JINNY / Prof. VISWANATHAN A / Prof. ARUMUGA ARUN R / Prof. VELU K**
Class Number(s) : **VL2025260501471/1480/1464/1438/5405/1478/1442/1461/6372/1486/1431/6755/1454/1445/1447/1458/1468**
Date of Examination : **30/01/2026**
Exam Duration : **90 minutes** **Maximum Marks: 50**

General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)
- Course Outcomes (Type the CO statements covered in this question paper. Use the CO number as per the syllabus copy)
  CO1: To analyze and implement basic programming constructs, decision-making statements, and loops in C/C++
  CO2: To apply modular programming techniques by utilizing functions, structures, and files for efficient problem-solving in real-world applications.

Q. No	Question	Module	Marks	CO	BL
1.	Write a C program to read an integer N and then get N integer elements for an **input_array**. Using appropriate conditional statements, looping constructs, and array operations, generate an **output_array** by performing the following operations: • If the array_element is a positive number, compute the factorial of the array_element and store the result in the corresponding position of the **output_array**. • If array_element is zero, compute the sum of all elements in the **input_array** and store the result in the corresponding position of the **output_array**. • If array_element is a negative number, compute the factorial of the square of |array_element| and store the result in the corresponding position of the **output_array**. • Finally, display the contents of the **input_array,** and **output_array.**	1	10	1	3

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: D1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST – I Key
WINTER SEMESTER 2025-2026

2.	List the different storage classes in C. Critically analyze how the selection of storage classes affects variable lifetime, scope, initialization, memory allocation, and overall program performance. With appropriate code-level illustrations, explain how an incorrect choice of storage class may result in logical errors, unnecessary memory consumption, or degraded execution efficiency.	1	10	1	4
3.	a) Write a C program that defines a user-defined function to receive an integer number from the main() function and reverse the digits of the number. The user-defined function must directly modify the same on the variable used in the main() function without using any return statement. Choose the suitable function calling method to achieve this requirement. Display the value of the integer number before and after invoking the function.	1	5	1	3
	b) Write a C program to read a string from the user and perform the following operations using standard string handling functions: • Determine and display the length of the string. • Check whether the given string is a palindrome.	1	5		
4.	a) Compare and contrast the dynamic memory allocation functions **malloc()** and **calloc()** in C with respect to their syntax, memory initialization behavior, and typical usage scenarios.	2	5	2	2

	b) Consider the following C program. Determine the exact output produced by the program. Then, clearly explain and differentiate the behavior and functionality of the expressions ***(p + 2), *(p) + 2, *p++,** and ***p - 2** with respect to pointer arithmetic, operator precedence, and value evaluation.
	
	`#include <stdio.h>`
	`int main() {`
	`  int a[5] = {10, 20, 30, 40, 50};`
	`  int *p = a;`
	`  printf("*(p + 2)  = %d\n", *(p + 2));`
	`  printf("*(p) + 2  = %d\n", *(p) + 2);`
	`  printf("*p++  = %d\n", *p++);`
	`  printf("*p - 2  = %d\n", *p - 2);`
	`  return 0;`
	`}`
		2	5		

5.	Develop a C program to read a two-dimensional matrix and generate its transpose using pointers to arrays rather than direct array indexing. Display both the original matrix and the transposed matrix. Subsequently, implement matrix multiplication between the original matrix and its transpose, and display the resulting matrix. The program should clearly demonstrate pointer-based traversal of 2D arrays.	2	10	2	3

****

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: D1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

Programme Name & Branch : **B.Tech & Common to all**
Course Code and Course Name : **BACSE104 - Structured and Object-Oriented Programming**
Faculty Name(s) : **Prof. SENTHIL KUMAR K / Prof. RAMESH BABU VEMULURI / Prof. VIJAYASHREE J /Prof. KEERTHIKA P / Prof. KALAIVANI K / Prof. GOUTAM MAJUMDER / Prof. DHIVYAA C R / Prof. SRI PREETHAA K R / Prof. GUNASEKAR M / Prof. SWETHA V / Prof. SYAMASUDHA VEERAGANDHAM / Prof. ASHOK KUMAR RAI / Prof. JABANJALIN HILDA J / Prof. VINILA JINNY / Prof. VISWANATHAN A / Prof. ARUMUGA ARUN R / Prof. VELU K**
Class Number(s) : **VL2025260501471/1480/1464/1438/5405/1478/1442/1461/6372/1486/1431/6755/1454/1445/1447/1458/1468**
Date of Examination : **30/01/2026**
Exam Duration : **90 minutes** **Maximum Marks: 50**

General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)
- Course Outcomes (Type the CO statements covered in this question paper. Use the CO number as per the syllabus copy)
  CO1: To analyze and implement basic programming constructs, decision-making statements, and loops in C/C++
  CO2: To apply modular programming techniques by utilizing functions, structures, and files for efficient problem-solving in real-world applications.

Q. No	Question	Module	Marks	CO	BL
1.	Write a C program to read an integer N and then get N integer elements for an **input_array**. Using appropriate conditional statements, looping constructs, and array operations, generate an **output_array** by performing the following operations: • If the array-element is a positive number, compute the factorial of the array-element and store the result in the corresponding position of the **output_array**. • If array-element is zero, compute the sum of all elements in the **input_array** and store the result in the corresponding position of the **output_array**. • If array-element is a negative number, compute the factorial of the square of |array-element| and store the result in the corresponding position of the **output_array**. • Finally, display the contents of the **input_array,** and **output_array.**	1	10	1	3

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

Key	#include <stdio.h>
	long long factorial(int num) {
	long long fact = 1;
	for (int i = 1; i <= num; i++) {
	fact *= i;
	}
	return fact;
	}
	int main() {
	int N, i;
	int input_array[100];
	long long output_array[100];
	long long sum = 0;
	printf("Enter N: ");
	scanf("%d", &N);
	printf("Enter %d elements:\n", N);
	for (i = 0; i < N; i++) {
	scanf("%d", &input_array[i]);
	sum += input_array[i];
	}
	for (i = 0; i < N; i++) {
	if (input_array[i] > 0) {
	output_array[i] = factorial(input_array[i]);
	}
	else if (input_array[i] == 0) {
	output_array[i] = sum;
	}
	else {
	int absVal = -input_array[i];
	int square = absVal * absVal;
	output_array[i] = factorial(square);
	}
	}
		printf("Input Array:\n");
		for (i = 0; i < N; i++) {
		printf("%d ", input_array[i]);
		}
		
		printf("\nOutput Array:\n");
		for (i = 0; i < N; i++) {
		printf("%lld ", output_array[i]);
		}
		return 0;
		}				

2.	List the different storage classes in C. Critically analyze how the selection of storage classes affects variable lifetime, scope, initialization, memory allocation, and overall program performance. With appropriate code-level illustrations, explain how an incorrect choice of storage class may result in logical errors, unnecessary memory consumption, or degraded execution efficiency.	1	10	1	4
Key	• Listing Storage Classes in C (1 Mark) • Impact and analysis of Storage Class Selection (5 Marks) • Code-Level Illustrations (4 Marks)				
3.	a) Write a C program that defines a user-defined function to receive an integer number from the main() function and reverse the digits of the number. The user-defined function must directly modify the same on the variable used in the main() function without using any return statement. Choose the suitable function calling method to achieve this requirement. Display the value of the integer number before and after invoking the function.	1	5	1	3

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

---

	b) Write a C program to read a string from the user and perform the following operations using standard string handling functions: • Determine and display the length of the string. • Check whether the given string is a palindrome.	1	5		

Key 3.a	`#include <stdio.h>`
	`void reverseNumber(int *num) {`
	`  int rev = 0, digit, temp;`
	`  temp = *num;`
	`  while (temp != 0) {`
	`  digit = temp % 10;`
	`  rev = rev * 10 + digit;`
	`  temp = temp / 10;`
	`  }`
	`  *num = rev;  // Modify the original variable`
	`}`
	`int main() {`
	`  int number;`
	`  printf("Enter an integer number: ");`
	`  scanf("%d", &number);`
	`  printf("Before reversing: %d\n", number);`
	`  reverseNumber(&number);  // Call by reference`
	`  printf("After reversing: %d\n", number);`
	`  return 0;`
	`}`				

Key	`#include <stdio.h>`
	`#include <string.h>`
	`int main() {`
	`  char str[100], rev[100];`
	`  int length;`
	`  printf("Enter a string: ");`
	`  scanf("%s", str);  // Reads a single word`
	`  /* Find length of the string */`
	`  length = strlen(str);`
	`  printf("Length of the string: %d\n", length);`
	`  /* Copy and reverse the string */`
	`  strcpy(rev, str);`
	`  strrev(rev);`
	`  /* Check palindrome using strcmp */`
	`  if (strcmp(str, rev) == 0) {`
	`  printf("The string is a palindrome.\n");`
	`  } else {`
	`  printf("The string is not a palindrome.\n");`
	`  }`
	`  return 0;`
	`}`				

4.	a) Compare and contrast the dynamic memory allocation functions **malloc()** and **calloc()** in C with respect to their syntax, memory initialization behavior, and typical usage scenarios.	2	5	2	2

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

---

	b) Consider the following C program. Determine the exact output produced by the program. Then, clearly explain and differentiate the behavior and functionality of the expressions ***(p + 2), *(p) + 2, *p++,** and ***p - 2** with respect to pointer arithmetic, operator precedence, and value evaluation.
	
	`#include <stdio.h>`
	`int main() {`
	`  int a[5] = {10, 20, 30, 40, 50};`
	`  int *p = a;`
	`  printf("*(p + 2) = %d\n", *(p + 2));`
	`  printf("*(p) + 2 = %d\n", *(p) + 2);`
	`  printf("*p++ = %d\n", *p++);`
	`  printf("*p - 2 = %d\n", *p - 2);`
	`  return 0;`
	`}`
		2	5		

Key 4.a

Aspect	malloc()	calloc()
Header file	`<stdlib.h>`	`<stdlib.h>`
Function prototype	`void *malloc(size_t size);`	`void *calloc(size_t n, size_t size);`
Number of arguments	One	Two
Memory size specification	Total size in bytes	Number of elements × size of each
Memory initialization	Not initialized (contains garbage values)	Initialized to zero
Speed	Faster (no initialization overhead)	Slightly slower due to zero-initialization
Safety	Risk of using uninitialized memory	Safer for immediate use
Typical usage	Single block allocation	Array allocation
Memory release	`free()`	`free()`

Key 4.b

Correct Program Output (2.5 marks)

```
*(p + 2)  = 30
*(p) + 2  = 12
*p++  = 10
*p - 2    = 18
```

Justification of elements: 2.5 Marks

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

5.	Develop a C program to read a two-dimensional matrix and generate its transpose using pointers to arrays rather than direct array indexing. Display both the original matrix and the transposed matrix. Subsequently, implement matrix multiplication between the original matrix and its transpose, and display the resulting matrix. The program should clearly demonstrate pointer-based traversal of 2D arrays.	2	10	2	3

Key

```
#include <stdio.h>
int main() {
   int r, c, i, j, k;
   printf("Enter rows and columns: ");
   scanf("%d %d", &r, &c);
   int a[10][10], t[10][10], result[10][10];
   int *p = &a[0][0];   // Pointer to first element of matrix
   int *pt = &t[0][0];
   int *pr = &result[0][0];
   /* Read matrix using pointer */
   printf("Enter matrix elements:\n");
   for (i = 0; i < r; i++) {
      for (j = 0; j < c; j++) {
         scanf("%d", (p + i * c + j));
      }
   }
   /* Transpose using pointer arithmetic */
   for (i = 0; i < r; i++) {
      for (j = 0; j < c; j++) {
         *(pt + j * r + i) = *(p + i * c + j);
      }
   }
   /* Display original matrix */
   printf("\nOriginal Matrix:\n");
   for (i = 0; i < r; i++) {
      for (j = 0; j < c; j++) {
         printf("%d ", *(p + i * c + j));
      }
      printf("\n");
   }
   /* Display transpose matrix */
   printf("\nTranspose Matrix:\n");
   for (i = 0; i < c; i++) {
      for (j = 0; j < r; j++) {
         printf("%d ", *(pt + i * r + j));
      }
      printf("\n");
   }
   /* Matrix multiplication: A × A^T */
```

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
WINTER SEMESTER 2025-2026

```
for (i = 0; i < r; i++) {
   for (j = 0; j < r; j++) {
      *(pr + i * r + j) = 0;
      for (k = 0; k < c; k++) {
         *(pr + i * r + j) +=
            (*(p + i * c + k)) * (*(pt + j * r + k));
      }
   }
}
/* Display result matrix */
printf("\nResult Matrix (A × A^T):\n");
for (i = 0; i < r; i++) {
   for (j = 0; j < r; j++) {
      printf("%d ", *(pr + i * r + j));
   }
   printf("\n");
}
return 0;
                                                    }
```

****