APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

KTU SPECIAL

We can together dream the B.tech

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

• KTU STUDY MATERIALS	• SYLLABUS
• KTU LIVE NOTIFICATION	• SOLVED QUESTION PAPERS

JOIN WITH US

WWW.KTUSPECIAL.IN KTUSPECIAL t.me/ktuspecial1

Reg No.:_____ Name:_____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

B.Tech Degree S1 (Challenge Course) Examination December 2024

Course Code: GXEST204

Course Name: PROGRAMMING IN C

Max. Marks: 60 Duration: 2 hours 30 minutes

PART A

(Answer all questions. Each question carries 3 marks) CO Marks

1 Identify the valid and invalid identifiers in C, given below. Justify your answer. 1 (3)

i) mark1 ii) _123 iii) $price

iv) rank list v) Xyz vi) -name

2 What will be the output of the following C program. Justify your answer. 1 (3)

```
void main() {
    int i=1234, j=0;
    while(i>0) {
       i/=10; j++;
    }
    printf("%d %d", i,j); }
```

3 What are the different ways to declare and initialize a single-dimensional array? 2 (3)

4 Write a C program to find length of a given string without using any string functions. 2 (3)

5 Compare the difference between structure and union data types in C. 3 (3)

6 What will be the output of the following C function, if the function was called 5 times in the main program? Justify your answer. 3 (3)

```
void fun() {
    static int count = 0;
    count++;
    printf("Count: %d\n", count); }
```

7 Write function prototypes of array of pointers, pointer to function and pointer to structure. 4 (3)

8		What happens in the following scenarios when using the statement fp = fopen("data.txt", "w");? • If data.txt does not exist on the disk. • If data.txt already exists on the disk.	5	(3)

PART B

(Answer any one full question from each module, each question carries 9 marks)

Module -1

9	a)	What will be the output of the following C program? Justify your answer.
		#include<stdio.h>
		void main() {
		int a=5, b=8, c=9, d;
		d=a++ + --b + c++;
		printf("%d %d %d %d\n", a, b, c, d); }
			1	(3)

	b)	Write a C program to check whether the given number is Armstrong number or not. [Hint: An Armstrong number is a number with the sum of, digits raised to the power of number of digits, returns the number. Example: $153 = 1^3 + 5^3 + 3^3$]	1	(6)
10	a)	Write a C program using bitwise operators to count the number of 1's in the binary representation of the given number.	1	(4)
	b)	Write a C program to check whether a given number is prime or not.	1	(5)

Module -2

11	a)	A set of N numbers are stored in an integer array. Write a C program to find the sum of odd numbers present in the array.	2	(4)
	b)	A set of N numbers are stored in an array. Write a C program to arrange the numbers in descending order using bubble sort algorithm.	2	(5)
12	a)	Write a C program to find the number of words and lines in a given string.	2	(5)
	b)	Write a C program to check whether the given string is palindrome or not without using any string manipulation function.	2	(4)

Module -3

13	a)	Write a C function to evaluate the series $1 + x + x^2 + \ldots\ldots + x^n$, given x and n.	3	(5)

	b)	What are recursive functions? Write a recursive C function to find the GCD of two positive numbers.	3	(4)
14	a)	Define a structure named Student with members roll_no, name, total_mark, percentage & grade. Create 10 student records, input roll_no, name & total_mark (out of 500) only. Write a C functions to evaluate and update percentage & grade of every student. Grades are calculated as follows. grade='O', if percentage >= 90, grade='A', if percentage >= 80 and <90 grade='B', if percentage >= 70 and <80 grade='C', if percentage >= 60 and <70 grade='D', if percentage >= 50 and <60 grade='E', if percentage >= 40 and <50 grade='F', if percentage < 40	3	(6)
	b)	Write a C function to display the details of all students with grade='O' in the above student records.	3	(3)

Module -4

15	a)	What is the difference between the malloc and calloc functions?	4	(3)
	b)	Write a C function to concatenate two strings using pointers?	4	(6)
16	a)	Write a C program to create a text file (input text through keyboard). Display the contents and size of the file created.	5	(4)
	b)	A stock file contains item_code, item_name, quantity and reorder_level. Write a C program to create a stock file of N items. Write a function to list items with quantity less than reorder_level.	5	(5)

*****

SCHEME OF VALUATION / ANSWER KEY

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
B.Tech Degree S1 (Challenge Course) Examination December 2024

Course Code:GXEST204

Course Name PROGRAMMING IN C

Max. Marks: 60			Duration: 2hours30minutes

General Instruction: For programming questions any programs with correct logic and syntax may be given full marks.

PART A

Identifier	Valid/Invalid	Reason
`mark1`	**Valid**	Contains only letters and digits, starts with a letter.
`_123`	**Valid**	Starts with `_` and contains valid characters.
`$price`	**Invalid**	`$` is not allowed in standard C identifiers.
`rank list`	**Invalid**	Contains a space, which is not allowed.
`Xyz`	**Valid**	Contains only letters and starts with a letter.
`-name`	**Invalid**	Starts with `-`, which is not allowed.

		(Answer all questions. Each question carries 3 marks)	CO	Marks
1		0.5 marks each	1	(3)
2		Output: - 1.5 marks i = 0 , j = 4 Justification: i becomes 0 and j contains number of digits in the number) 1.5 marks	1	(3)
3		**Method** / **Example** / **Result** Full Initialization / int arr[5] = {1, 2, 3, 4, 5}; / {1, 2, 3, 4, 5} Partial Initialization / int arr[5] = {1, 2}; / {1, 2, 0, 0, 0} Omit Size / int arr[] = {1, 2, 3}; / Size determined Dynamic Initialization / arr[i] = i * 2; / Values assigned	2	(3)

4		#include <stdio.h>
		void main() {
		char str[100];
		int length = 0;
			2	(3)

```
printf("Enter a string: ");
fgets(str, sizeof(str), stdin); // Read string input including spaces
// Loop to count characters until the null terminator ('\0') is reached
while (str[length] != '\0') {
    length++;
}
// Adjust length if fgets adds a newline character
if (str[length - 1] == '\n') {
    length--;
}
printf("The length of the string is: %d\n", length);
}
```

Feature	Structure	Union
Memory Allocation	Each member has its own memory space. The total memory required is the sum of the memory sizes of all members.	All members share the same memory location. The total memory required is equal to the size of the largest member.
Accessing Members	All members can be accessed simultaneously, as each member has its own memory space.	Only one member can be accessed at a time, as all members share the same memory space.
Usage	Used when you need to store multiple values of different types simultaneously.	Used when you need to store one of many values, but only one at a time, saving memory.
Data Sharing	Members do not share data. Each member has its own memory location.	Members share the same memory location, which means they overwrite each other.
Example	`struct Person {int age; float height;};`	`union Data {int i;float f;};`

5 Any THREE differences shown below. – 1 mark each 3 (3)

6		<u>Output:</u> - 1.5 marks Count: 1 Count: 2 Count: 3 Count: 4 Count: 5 <u>Justification:</u> Since count is a static variable, it is initialized once and retains its value across all calls to the fun() function. With each call, the function increments the value of count, and this updated value is used in the next call. - 1.5 marks	3	(3)
7		**Array of Pointers**:: void function_name(int ***arr[]**, int size); **Pointer to Function**: void function_name(int (***func_ptr)**(int, int)); **Pointer to Structure**: void function_name(**struct Person *person_ptr**); 1 mark each	4	(3)
8		1. **If `data.txt` does not exist on the disk**: o The file `data.txt` is created on the disk. o The file pointer `fp` points to the newly created file. o If the file cannot be created (e.g., due to insufficient permissions), `fp` will be `NULL`. 2. **If `data.txt` already exists on the disk**: o The file `data.txt` is opened, but its content is truncated (i.e., all existing data is erased). o The file pointer `fp` points to the empty file ready for writing.	5	(3)
		PART B		
		(Answer any one full question from each module, each question carries 9 marks)		
		Module -1		
9	a)	<u>Output:</u> - 2 marks 6 7 10 21 <u>Justification:</u> 1 mark `a++` evaluates to `5` (then `a` becomes `6`). `--b` evaluates to `7` (then `b` becomes `7`). `c++` evaluates to `9` (then `c` becomes `10`). The final value of `d` is `21`.	1	(3)
	b)	#include <stdio.h> #include <math.h>	1	(6)

```
void main() {
    int num, temp, remainder, sum = 0, digits = 0;

    // Read the number from the user
    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;

    // Calculate the number of digits in the given number
    while (temp != 0) {
        temp /= 10;
        digits++;
    }

    temp = num;

    // Calculate the sum of the powers of the digits
    while (temp != 0) {
        remainder = temp % 10;  // Extract the last digit
        sum += pow(remainder, digits);  // Add the power of the digit to sum
        temp /= 10;  // Remove the last digit
    }

    // Check if the number is an Armstrong number
    if (sum == num) {
        printf("%d is an Armstrong number.\n", num);
    } else {
        printf("%d is not an Armstrong number.\n", num);
    }
}
```

10	a)	#include <stdio.h>
		int countOnes(int num) {
		int count = 0;
		while (num != 0) {
		// Check the least significant bit
		if (num & 1) {
		count++;
		}
		// Right shift the number by 1
		num >>= 1;
		}
		return count;
		}
		
		void main() {
			1	(4)

```
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    int result = countOnes(number);
    printf("The number of 1s in the binary representation of %d is: %d\n",
number, result);
}
```

b)

```
#include <stdio.h>

void main() {
        int num, i, isPrime = 1;

        // Read the number from the user
        printf("Enter a number: ");
        scanf("%d", &num);

        // Check if the number is less than 2 (not prime)
        if (num <= 1) {
                isPrime = 0;  // Numbers less than or equal to 1 are not prime
        } else {
                // Check for factors from 2 to sqrt(num)
                for (i = 2; i <= num / 2; i++) {
                    if (num % i == 0) {
                        isPrime = 0;  // If a factor is found, the number is not
prime
                        break;
                    }
                }
        }

        // Print the result
        if (isPrime) {
            printf("%d is a prime number.\n", num);
        } else {
            printf("%d is not a prime number.\n", num);
        }
}
```

1 (5)

Module -2

11 a)

```
#include <stdio.h>

void main() {
    int N, sum = 0;

    // Read the size of the array
    printf("Enter the number of elements in the array: ");
```

2 (4)

```
    scanf("%d", &N);

    int arr[N];  // Declare an array of size N

    // Read the elements of the array
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    // Iterate through the array and sum up the odd numbers
    for (int i = 0; i < N; i++) {
        if (arr[i] % 2 != 0) {  // Check if the number is odd
            sum += arr[i];  // Add the odd number to sum
        }
    }

    // Print the sum of odd numbers
    printf("Sum of odd numbers in the array: %d\n", sum);
}
```

b)

2

(5)

```
#include <stdio.h>
void main() {
    int N,;

    // Read the number of elements in the array
    printf("Enter the number of elements in the array: ");
    scanf("%d", &N);

    int arr[N], i, j, temp;;

    // Read the elements of the array
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    //bubbleSort descending
     for (i = 0; i < N - 1; i++) {
        for (j = 0; j < N - i - 1; j++) {
            // Compare adjacent elements
            if (arr[j] < arr[j + 1]) {
                // Swap the elements if they are in the wrong order
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
```

		// Print the array after sorting
		printf("Array sorted in descending order:\n");
		for (int i = 0; i < N; i++) {
		printf("%d ", arr[i]);
		}
		printf("\n");
		}		

12	a)	#include <stdio.h>
		#include <ctype.h>
		
		void main() {
		char str[1000];
		int i = 0, words = 0, lines = 0;
		
		// Read the input string (with spaces, and user can press Enter to enter multiple lines)
		printf("Enter a string (press Enter and Ctrl+D to end input):\n");
		while (fgets(str, sizeof(str), stdin)) {
		// Increment lines counter whenever a new line is read
		lines++;
		
		// Loop through each character of the line
		for (i = 0; str[i] != '\0'; i++) {
		// If the character is a space or newline, we consider the previous word ended
		if (isspace(str[i])) {
		if (i > 0 && !isspace(str[i - 1])) {
		words++; // A word ends when a space or newline follows a non-space character
		}
		}
		}
		
		// If the last character in the string is a non-space character, it is counted as a word
		if (i > 0 && !isspace(str[i - 1])) {
		words++;
		}
		}
		printf("Total number of words: %d\n", words);
		printf("Total number of lines: %d\n", lines);
		}
			2	(5)

	b)	#include <stdio.h>
		
		void main() {
		char str[100], reverseStr[100];
		int length = 0, i, j;
		
		// Read the input string
		printf("Enter a string: ");
		fgets(str, sizeof(str), stdin);
		
		// Find the length of the string
		while (str[length] != '\0' && str[length] != '\n') {
		length++;
		}
		
		// Reverse the string manually
		for (i = 0, j = length - 1; j >= 0; i++, j--) {
		reverseStr[i] = str[j];
		}
		reverseStr[i] = '\0'; // Add null terminator to the reversed string
		
		// Compare original string and reversed string
		int isPalindrome = 1; // Assume it is a palindrome unless proven otherwise
		for (i = 0; i < length; i++) {
		if (str[i] != reverseStr[i]) {
		isPalindrome = 0; // If characters don't match, it's not a palindrome
		break;
		}
		}
		
		// Output the result
		if (isPalindrome) {
		printf("The given string is a palindrome.\n");
		} else {
		printf("The given string is not a palindrome.\n");
		}
		}
			2	(4)

Module -3				

13	a)	#include <stdio.h>
		
		// Function to calculate the sum of the series: 1 + x + x^2 + ... + x^n
		double evaluateSeries(int x, int n) {
		double sum = 1; // Initialize sum to 1 (for the first term)
		int term = 1; // To keep track of x^i for each term
		
		// Calculate the sum of the series
		for (int i = 1; i <= n; i++) {
			3	(5)

```
        term *= x;  // term = x^i, multiply previous term by x
        sum += term; // Add the current term to the sum
    }
    return sum;
}

void  main() {
    int x, n;

    // Input values for x and n
    printf("Enter the value of x: ");
    scanf("%d", &x);
    printf("Enter the value of n: ");
    scanf("%d", &n);

    // Call the function and display the result
    double result = evaluateSeries(x, n);
    printf("The result of the series 1 + x + x^2 + ... + x^n is: %.2f\n", result);
}
```

b) A **recursive function** is a function that calls itself to solve a smaller instance of the same problem. The function typically has: (3) (4)

1. **Base Case**: A condition that stops the recursion.
2. **Recursive Case**: A call to the same function with a smaller or simpler input. (1.5 marks)

```
// Recursive function to find GCD
int gcd(int a, int b) {
    if (b == 0) {
        return a; // Base case: GCD is 'a' when 'b' becomes 0
    }
    return gcd(b, a % b); // Recursive case                    (2.5 marks)
}
```

14 a)

```
#include <stdio.h>

// Define the structure Student
struct Student {
    int roll_no;
    char name[50];
    int total_mark;
    float percentage;
    char grade;
};

// Function to calculate and update percentage and grade

void evaluateGrade(struct Student* student) {
    // Calculate percentage
```

3 (6)

```
    student->percentage = (float)student->total_mark / 500 * 100;

    // Assign grade based on percentage
    if (student->percentage >= 90) {
        student->grade = 'O';
    } else if (student->percentage >= 80) {
        student->grade = 'A';
    } else if (student->percentage >= 70) {
        student->grade = 'B';
    } else if (student->percentage >= 60) {
        student->grade = 'C';
    } else if (student->percentage >= 50) {
        student->grade = 'D';
    } else if (student->percentage >= 40) {
        student->grade = 'E';
    } else {
        student->grade = 'F';
    }
}

void main() {
    struct Student students[10];

    // Input student records
    for (int i = 0; i < 10; i++) {
        printf("\nEnter details for Student %d:\n", i + 1);
        printf("Roll No: ");
        scanf("%d", &students[i].roll_no);
        getchar();  // To consume the newline left by scanf
        printf("Name: ");
        fgets(students[i].name, sizeof(students[i].name), stdin);
        // Remove newline character from the name
        students[i].name[strcspn(students[i].name, "\n")] = '\0';
        printf("Total Marks (out of 500): ");
        scanf("%d", &students[i].total_mark);

        // Call the function to evaluate percentage and grade
        evaluateGrade(&students[i]);
    }

    // Display the results
    printf("\nStudent Report:\n");
    printf("Roll No\tName\t\tTotal Marks\tPercentage\tGrade\n");
    for (int i = 0; i < 10; i++) {
        printf("%d\t%s\t\t%d\t\t%.2f%%\t\t%c\n",
               students[i].roll_no, students[i].name, students[i].total_mark,
               students[i].percentage, students[i].grade);
    }
}
```

	b)	// Function to display details of students with grade 'O'
		void displayGradeOStudents(struct Student students[], int n) {
		printf("\nStudents with grade 'O':\n");
		printf("Roll No\tName\t\tTotal Marks\tPercentage\tGrade\n");
		int found = 0;
		
		// Loop through the array of students and check for grade 'O'
		for (int i = 0; i < n; i++) {
		if (students[i].grade == 'O') {
		// Display the student's details if the grade is 'O'
		printf("%d\t%s\t\t%d\t\t%.2f%%\t\t%c\n",
		students[i].roll_no, students[i].name, students[i].total_mark,
		students[i].percentage, students[i].grade);
		found = 1;
		}
		}
		}
		
		// If no student with grade 'O' is found
		if (!found) {
		printf("No students with grade 'O'.\n");
		}
		}
			3	(3)

		Module -4		
15	a)	□ **Memory Initialization**: • malloc: Allocates memory but does not initialize it. The allocated memory contains garbage values. • calloc: Allocates memory and initializes all bytes to zero. □ **Syntax**: • malloc: void* malloc(size_t size); Example: int *arr = (int*)malloc(5 * sizeof(int)); • calloc: void* calloc(size_t num, size_t size); Example: int *arr = (int*)calloc(5, sizeof(int)); □ **Parameters**: • malloc: Takes a single parameter, which is the total number of bytes to allocate. • calloc: Takes two parameters: the number of blocks (num) and the size of each block (size). □ **Performance**: • malloc is faster because it does not initialize memory. • calloc is slower due to the additional step of initializing memory to zero. □ **Use Case**: • Use malloc when memory initialization is not required. • Use calloc when you need memory initialized to zero. (At least three differences – 3 marks)	4	(3)

	b)	void concatenateStrings(char *str1, const char *str2) {
		// Move the pointer to the end of the first string
		while (*str1 != '\0') {
		str1++;
		}
		// Append characters from the second string to the first string
		while (*str2 != '\0') {
		*str1 = *str2;
		str1++;
		str2++;
		}
		// Add the null terminator at the end
		*str1 = '\0';
		}
			4	(6)

16	a)	#include <stdio.h>
		#include <stdlib.h>
		
		void main() {
		FILE *file;
		char str[1000];
		long file_size;
		
		// Create and open the file for writing
		file = fopen("example.txt", "w");
		if (file == NULL) {
		printf("Error opening file for writing!\n");
		return 1;
		}
		
		// Input text from the user
		printf("Enter text to write to the file (Press Enter to finish):\n");
		fgets(str, sizeof(str), stdin); // Read the input text
		
		// Write the text to the file
		fputs(str, file);
		
		// Close the file after writing
		fclose(file);
		
		// Reopen the file for reading and checking its size
		file = fopen("example.txt", "r");
		if (file == NULL) {
			5	(4)

```
    printf("Error opening file for reading!\n");
    return 1;
}

// Display the contents of the file
printf("\nContents of the file:\n");
while (fgets(str, sizeof(str), file) != NULL) {
    printf("%s", str);
}

// Get the size of the file using ftell()
fseek(file, 0, SEEK_END);  // Move file pointer to the end
file_size = ftell(file);   // Get the current position (size of the file)
printf("\nSize of the file: %ld bytes\n", file_size);

// Close the file
fclose(file);
}
```

b)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define a structure to hold stock item details
struct StockItem {
    int item_code;
    char item_name[50];
    int quantity;
    int reorder_level;
};

// Function to create stock file and input details of N items
void createStockFile(int n) {
    FILE *file = fopen("stock.dat", "wb");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }

    struct StockItem item;
    for (int i = 0; i < n; i++) {
        printf("Enter details for item %d:\n", i + 1);
        printf("Item Code: ");
        scanf("%d", &item.item_code);
        getchar();  // To consume the newline character left by scanf
        printf("Item Name: ");
        fgets(item.item_name, sizeof(item.item_name), stdin);
        item.item_name[strcspn(item.item_name, "\n")] = '\0';    // Remove
newline character from name
        printf("Quantity: ");
```

5 (5)

```
        scanf("%d", &item.quantity);
        printf("Reorder Level: ");
        scanf("%d", &item.reorder_level);

        // Write the item to the file
        fwrite(&item, sizeof(struct StockItem), 1, file);
    }

    fclose(file);
    printf("\nStock file created successfully.\n");
}

// Function to list items where quantity is less than reorder level
void listItemsBelowReorderLevel() {
    FILE *file = fopen("stock.dat", "rb");
    if (file == NULL) {
        printf("Error opening file for reading.\n");
        return;
    }

    struct StockItem item;
    int found = 0;
    printf("\nItems with quantity below reorder level:\n");
    printf("---------------------------------------------\n");
    printf("Item Code | Item Name          | Quantity | Reorder Level\n");
    printf("---------------------------------------------\n");
    while (fread(&item, sizeof(struct StockItem), 1, file)) {
        if (item.quantity < item.reorder_level) {
            printf("%-10d|    %-23s|    %-8d|    %-14d\n",    item.item_code,
item.item_name, item.quantity, item.reorder_level);
            found = 1;
        }
    }

    if (!found) {
        printf("No items found with quantity below reorder level.\n");
    }

    fclose(file);
}

void main() {
    int n;

    // Create stock file with N items
    printf("Enter the number of items: ");
    scanf("%d", &n);
    createStockFile(n);
```

		// List items with quantity below reorder level
		listItemsBelowReorderLevel();
		}		