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

MODEL QUESTION PAPER

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SECOND SEMESTER B.TECH DEGREE 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		Convert the below if-else to conditional operator
		`if (first_check)`
		`{`
		`"access denied";`
		`}`
		`else`
		`{`
		`if (second_check)`
		`{`
		`"access denied";`
		`}`
		`else`
		`{`
		`"access granted";`
		`}`
		`}`
			1	(3)

2		State proper reasons to identify whether the given identifiers are valid or not. i) 9marks ii) Mark 2020 iii) v4vote iv) _csdept v) @cse vi) AxB	1	(3)
3		Explain the different ways in which you can declare and initialize a single-dimensional array.	2	(3)
4		How can a list of strings be stored within a two-dimensional array? How can the individual strings be processed? What library functions are available to simplify string processing?	2	(3)
5		Suppose function F1 calls function F2 within a C program. Does the order of the function definitions make any difference? Explain.	3	(3)

6		Identify the storage class in the below code?
		`#include<stdio.h>`
		`int a;`
		`int main()`
		`{`
		`printf("a=%d",a);`
		`return 0;`
		`}`
			5	(3)

7		Suppose a function receives a pointer as an argument. Explain how the function	4	(3)

		prototype is written. In particular, explain how the data type of the pointer argument is represented.		
8		While using the statement, fp = fopen("myfile.c", "r") ; what happens if, − ‘myfile.c’ does not exist on the disk − ‘myfile.c’ exists on the disk	5	(3)
PARTB				
Answer any one full question from each module. Each question carries 9 marks				
Module 1				
9	a	Write a C Program to check if a given number is a strong number or not. A strong number is a number in which the sum of the factorial of the digits is equal to the number itself. Eg:- 145=1!+4!+5!=1+24+120=145	1	(5)
	b	Write a C program that calculates and prints the result of the following expression: result = 5 * (6 + 2) / 4 - 3. Explain how parentheses affect the precedence of operators in this expression.	1	(4)
10	a	Write a C program that reads employee's ID, name, status (‘F’ for full-time and ‘P’ for part-time) and basic pay. DA and HRA for full-time employees are 75% and 10% of basic pay respectively. Part-time employees have basic pay only. Print the total salary of the employee.	1	(6)
	b	Explain the difference between primitive (basic) data types and derived data types in C. Provide examples for each category. Why is it important to choose the correct data type for variables in a program?	1	(3)
Module 2				
11	a	Write a C program to reverse the content of an array without using another array.	2	(4)
	b	Given a list of cities, their literacy level, and the population, write a program to print the list cities sorted lexicographically. Assume duplication in city names.. Break the ties with the value of literacy level, and then the population.	2	(5)
12	a	Write a C program to rotate (re-insert rightmost element at the left) elements of an array to the right by n-steps.	2	(4)
	b	Write a C program that allows the user to enter a line of text, store it in an array, and then display it backward. The line length should be unspecified (terminated by pressing the Enter key), but assume it will not exceed 80 characters.	2	(5)
Module 3				
13	a	Develop a function that takes the two dates as input and compares them. The function should return 1 if date1 is earlier than date2, return 0 if date1 is later than date2. Write code to display whether date1 is earlier, later, or the same as date2 based on the function’s result.	3	(4)
	b	Define a structure named Vehicle with members model (character array), year (integer), and price (float). Create an array of 10 Vehicle structures and write a function to accept user input for each vehicle's details and Sort the array in descending order based on price.	2	(5)
14	a	Define a structure that can describe a hotel with the following members: name, address, grade, average room charge, and number of rooms. Write a function to print out hotels of a given grade, sorted in order of their average room charges.	3	(4)
	b	Write a function to print out hotels with an average room charge less than a specified value in the above problem.	3	(5)

Module 4				
15	a	Write a c program snippet to open a file in write mode and check if the file opened successfully. if it fails ,print an error message and terminate the program.	5	(4)
	b	Two persons want to access a file "sample.txt". First person want to read the data from the file. The second person want to read and write the data from and to the file simultaneously. Can you help them to do so by writing the corresponding programming codes?	5	(5)
16	a	Write a function calculates the roots of a quadratic equation with coefficients *a*, *b*, and *c* (passed as pointer parameters).	4	(4)
	b	In a small firm, employee numbers are given in serial numerical order, that is 1, 2, 3, etc. − Create a file of employee data with following information: employee number, name, sex, gross salary. − If more employees join, append their data to the file. − If an employee with serial number 25 (say) leaves, delete the record by making gross salary 0. − If some employee's gross salary increases, retrieve the record and update the salary. Write a program to implement the above operations.	5	(5)
*****				

Programming in C - KTU 2024 Scheme Answers (Detailed)

1. Conditional Operator Conversion

The given code uses nested if-else conditions to decide access based on two checks. This can be replaced using a conditional (ternary) operator, which simplifies the code into a single line expression.

Original:

```
if (first_check) {
   "access denied";
} else {
   if (second_check) {
      "access denied";
   } else {
      "access granted";
   }
}
```

Using Conditional Operator:

```
(first_check) ? "access denied" : ((second_check) ? "access denied" : "access granted");
```

This means: if `first_check` is true, it returns "access denied", else it checks `second_check`, and again returns "access denied" if true, otherwise "access granted".

2. Identifier Validity

Identifiers are names used to identify variables, functions, arrays, etc. in C. Rules include:

- Must begin with a letter (A–Z or a–z) or an underscore (_)

- Can contain letters, digits (0–9), and underscores

- Cannot contain special characters like @, #, etc.

- Cannot be a C keyword

Identifier	Valid/Invalid	Reason
9marks	Invalid	Cannot start with a digit.

Mark 2020	Invalid	Contains a space.
v4vote	Valid	Starts with a letter and contains valid characters.
_csdept	Valid	Identifiers can start with an underscore.
@cse	Invalid	Contains invalid character `@`.
AxB	Valid	All characters are valid letters.

3. Declaring and Initializing Single-Dimensional Arrays

Arrays are used to store multiple values of the same data type in a single variable.
Single-dimensional arrays can be declared and initialized in the following ways:

1. Declaration only:
int arr[5]; // Memory is allocated but not initialized.

2. Declaration with size and initialization:
int arr[5] = {1, 2, 3, 4, 5}; // Array elements initialized.

3. Initialization without specifying size:
int arr[] = {1, 2, 3}; // Size is deduced from the number of elements.

4. Partial Initialization:
int arr[5] = {1, 2}; // First two elements initialized, others set to 0 by default.

4. String List in 2D Array and String Processing

In C, a list of strings can be stored using a 2D character array, where each row stores one string:

Example:
char names[3][20] = {"Alice", "Bob", "Charlie"};

Here, names is a 2D array storing 3 strings with a maximum of 19 characters each (1 for null character).

To access or process individual strings:
Use a loop:

```
for (int i = 0; i < 3; i++) {
    printf("%s\n", names[i]);
}
```

Library functions from <string.h> that simplify processing:
- strlen(): Returns length of the string
- strcpy(dest, src): Copies one string to another
- strcmp(str1, str2): Compares two strings
- strcat(dest, src): Concatenates strings
- strstr(haystack, needle): Finds a substring

5. Function Call Order in C

In C, the compiler needs to know the signature of a function before it is used.
If function F1 calls function F2, the order of their definitions matters only if no prototype is provided.

Two options:
1. Define F2 before F1:

```
void F2() { /*...*/ }
void F1() { F2(); }
```

2. Use a prototype before F1:

```
void F2();
void F1() { F2(); }
void F2() { /*...*/ }
```

Using prototypes allows flexibility in arranging function definitions in any order.

6. Storage Class in the Given Code

```
#include<stdio.h>
int a;  // Global variable

int main() {
    printf("a=%d",a);
    return 0;
}
```

Here, the variable `a` is declared outside any function, hence it is a global variable.

Its default storage class is `extern`.
Details:

- Storage Class: extern
- Storage Location: Global memory (Data segment)
- Scope: Throughout the program (all functions)
- Lifetime: Exists for the entire duration of the program

`extern` variables are automatically initialized to 0 if not explicitly initialized.

---

7. Suppose a function receives a pointer as an argument. Explain how the function prototype is written. In particular, explain how the data type of the pointer argument is represented.

Answer:

When a function receives a **pointer** as an argument, the function prototype must specify:

- The **data type** the pointer points to.
- That the argument is a **pointer**, by using the `*` symbol.

General Syntax of Function Prototype with Pointer:

```
return_type function_name(data_type *parameter_name);
```

Examples:

- Function accepting a pointer to an integer:
- `void modify(int *ptr);`

  This means `modify` is a function that takes a pointer to an integer.

- Function accepting a pointer to a character:
- `void printString(char *str);`

  Here, `str` is a pointer to a character, typically used for strings.

- Function prototype with multiple pointer arguments:
- `void swap(int *a, int *b);`

Explanation:

- The asterisk `*` before the variable name indicates that it is a pointer.

- The data type (like `int`, `char`, `float`) tells what type of data the pointer is pointing to.

---

8. While using the statement:

```
fp = fopen("myfile.c", "r");
```

What happens if:

---

a) 'myfile.c' does not exist on the disk

- The file **cannot be opened** because the `"r"` (read) mode only opens **existing files**.
- `fopen()` will return **NULL**.
- You should always check if the file pointer is `NULL` to handle such errors:
- `if (fp == NULL) {`
- `    printf("File not found or error opening file.\n");`
- `}`

---

b) 'myfile.c' exists on the disk

- The file is **successfully opened in read mode**.
- The file pointer `fp` will point to the **beginning** of the file.
- You can now perform **reading operations** such as `fgetc()`, `fgets()`, or `fread()`.

Summary of File Modes in `fopen()`:

Mode	Description
"r"	Read – File must exist
"w"	Write – Creates new file or overwrites existing
"a"	Append – Writes at end; creates if not exists
"r+"	Read & Write – File must exist
"w+"	Read & Write – Creates or overwrites file
"a+"	Read & Append – Creates if not exists

Module 1 – Question 9

(a) Strong Number Check:

Program:

```
#include <stdio.h>

int factorial(int n) {
    int fact = 1;
    for(int i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    int num, originalNum, remainder, sum = 0;

    printf("Enter a number: ");
    scanf("%d", &num);

    originalNum = num;
    while(num != 0) {
        remainder = num % 10;  // Get the last digit
        sum += factorial(remainder);  // Add factorial of the digit
        num /= 10;  // Remove the last digit
    }

    if(sum == originalNum) {
        printf("%d is a Strong number.\n", originalNum);
    } else {
        printf("%d is not a Strong number.\n", originalNum);
    }

    return 0;
}
```

Explanation:

- **Strong Number Definition:** A number is considered a "strong number" if the sum of the factorials of its digits is equal to the number itself. For example, 145 is a strong number because $1! + 4! + 5! = 145$.
- **Steps:**
  - The program calculates the factorial of each digit using a separate `factorial()` function.
  - It iterates over the digits of the number, computes the sum of the factorials, and compares this sum with the original number.
- **Input:** A number is entered by the user.
- **Output:** The program checks if the number is a strong number and prints the result.

(b) Expression Calculation with Parentheses:

Program:

```
#include <stdio.h>
int main() {
    float result;

    result = 5 * (6 + 2) / 4 - 3;

    printf("Result of the expression: %.2f\n", result);
    return 0;
}
```

Explanation:

- **Expression:** The expression is `5 * (6 + 2) / 4 - 3`.
- **Operator Precedence:**
  - Parentheses have the highest precedence, so `6 + 2` is evaluated first, which gives `8`.
  - Then, the multiplication `5 * 8` is evaluated, which results in `40`.
  - Next, the division `40 / 4` is computed, resulting in `10`.
  - Finally, `10 - 3` is evaluated, resulting in `7`.
- **Output:** The result of the expression is displayed as `7.00`.

Module 2 – Question 11

(a) Reverse the Content of an Array Without Using Another Array:

Program:

```
#include <stdio.h>

void reverseArray(int arr[], int n) {
    int start = 0, end = n - 1, temp;
    while(start < end) {
        // Swap elements
        temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}
```

```c
int main() {
    int arr[100], n;

    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    reverseArray(arr, n);

    printf("Reversed Array: ");
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Explanation:

- **Reverse Logic:** The program uses a two-pointer approach to reverse the array in place:
    - One pointer starts from the beginning (`start`), and the other starts from the end (`end`).
    - The elements at these two positions are swapped, and the pointers move towards each other until they meet in the middle.
- **Input:** The user provides the number of elements and the elements themselves.
- **Output:** The array is printed after being reversed.

(b) Sort Cities Based on Name, Literacy, and Population:

Program:

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

struct City {
    char name[50];
    float literacy;
    int population;
};

int compareCities(const void *a, const void *b) {
    struct City *cityA = (struct City *)a;
    struct City *cityB = (struct City *)b;

    int nameComparison = strcmp(cityA->name, cityB->name);
    if(nameComparison != 0) {
```

```
        return nameComparison;
    }

    if(cityA->literacy < cityB->literacy) return 1;
    else if(cityA->literacy > cityB->literacy) return -1;

    if(cityA->population < cityB->population) return 1;
    else return -1;
}

int main() {
    struct City cities[100];
    int n;

    printf("Enter number of cities: ");
    scanf("%d", &n);

    for(int i = 0; i < n; i++) {
        printf("Enter city %d details:\n", i + 1);
        printf("City Name: ");
        scanf("%s", cities[i].name);
        printf("Literacy Level: ");
        scanf("%f", &cities[i].literacy);
        printf("Population: ");
        scanf("%d", &cities[i].population);
    }

    qsort(cities, n, sizeof(struct City), compareCities);

    printf("\nCities sorted lexicographically, then by literacy, and
then by population:\n");
    for(int i = 0; i < n; i++) {
        printf("%s %.2f %d\n", cities[i].name, cities[i].literacy,
cities[i].population);
    }

    return 0;
}
```

Explanation:

- **Sorting Logic:** The cities are first compared lexicographically (alphabetically) using `strcmp`. If the names are the same, then the cities are compared based on literacy levels in descending order. If literacy levels are also equal, the cities are compared based on population in descending order.
- **Input:** The user enters the details of each city (name, literacy level, and population).
- **Output:** The cities are printed after being sorted based on the criteria.

Module 3 – Question 13

(a) Compare Two Dates:

Program:

```
#include <stdio.h>

struct Date {
    int day, month, year;
};

int compareDates(struct Date date1, struct Date date2) {
    if(date1.year < date2.year) return 1;   // date1 is earlier
    else if(date1.year > date2.year) return 0;  // date1 is later
    if(date1.month < date2.month) return 1;   // date1 is earlier
    else if(date1.month > date2.month) return 0;  // date1 is later
    if(date1.day < date2.day) return 1;    // date1 is earlier
    else if(date1.day > date2.day) return 0;   // date1 is later
    return -1;   // Both dates are same
}

int main() {
    struct Date date1, date2;

    printf("Enter first date (dd mm yyyy): ");
    scanf("%d %d %d", &date1.day, &date1.month, &date1.year);

    printf("Enter second date (dd mm yyyy): ");
    scanf("%d %d %d", &date2.day, &date2.month, &date2.year);

    int result = compareDates(date1, date2);

    if(result == 1) {
        printf("Date1 is earlier than Date2\n");
    } else if(result == 0) {
        printf("Date1 is later than Date2\n");
    } else {
        printf("Both dates are the same\n");
    }

    return 0;
}
```

Explanation:

- **Structure Definition:** A `Date` structure holds day, month, and year.
- **Comparison Logic:** The function `compareDates()` compares two dates by year, month, and day.
    - It returns `1` if `date1` is earlier, `0` if `date1` is later, and `-1` if both dates are the same.
- **Input:** The user enters two dates (day, month, and year).

- **Output:** The program prints whether the first date is earlier, later, or the same as the second.

Module 1 – Question 10

a. C Program for Employee Salary Calculation

```
#include <stdio.h>

int main() {
    int emp_id;
    char name[50], status;
    float basic, da = 0, hra = 0, total;

    printf("Enter Employee ID: ");
    scanf("%d", &emp_id);

    printf("Enter Name: ");
    scanf("%s", name);

    printf("Enter Status (F for Full-time, P for Part-time): ");
    scanf(" %c", &status);

    printf("Enter Basic Pay: ");
    scanf("%f", &basic);

    if (status == 'F' || status == 'f') {
        da = 0.75 * basic;
        hra = 0.10 * basic;
    }

    total = basic + da + hra;

    printf("\n--- Employee Details ---\n");
    printf("ID: %d\nName: %s\nStatus: %c\nTotal Salary: %.2f\n",
emp_id, name, status, total);

    return 0;
}
```

Explanation:

- If the employee is **full-time**, we calculate:
  - **DA** = 75% of basic pay
  - **HRA** = 10% of basic pay
- **Part-time** employees only receive the basic pay.
- The program reads details and calculates total salary accordingly.

---

b. Primitive vs Derived Data Types in C

Primitive Data Types	Derived Data Types
Basic building blocks of data	Formed from primitive types
Examples: `int`, `float`, `char`	Examples: `array`, `pointer`, `struct`
Used for simple data	Used for complex data and relations

Importance of Choosing the Right Data Type:

- Ensures **efficient memory usage**
- Prevents **type errors**
- Enhances **code clarity and performance**

Module 2 – Question 12

a. Program to Rotate Array to the Right by One Step

```
#include <stdio.h>

int main() {
    int arr[100], n, last;

printf("Enter number of elements: ");
scanf("%d", &n);

printf("Enter elements: ");
for(int i = 0; i < n; i++)
    scanf("%d", &arr[i]);

last = arr[n - 1];

for(int i = n - 1; i > 0; i--)
    arr[i] = arr[i - 1];

arr[0] = last;

printf("Array after rotation: ");
for(int i = 0; i < n; i++)
    printf("%d ", arr[i]);

    return 0;
}
```

Explanation:

- The last element is stored temporarily.
- All elements are shifted one position to the right.
- The last element is placed at the first position.

b. Program to Display Line Backward

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

int main() {
    char text[81];

    printf("Enter a line of text: ");
    fgets(text, sizeof(text), stdin);

    int len = strlen(text);

    printf("Reversed text: ");
    for (int i = len - 2; i >= 0; i--)  // -2 to exclude newline
        printf("%c", text[i]);

    return 0;
}
```

Explanation:

- Uses `fgets()` to read input up to 80 characters.
- String is reversed by iterating from the end to the beginning.

Module 3 – Question 14

a. Hotel Structure and Sorting Based on Average Room Charge

Objective:

- Create a structure to store hotel details.
- Filter and print hotels based on **grade**.
- Sort filtered hotels by **average room charge** in ascending order.

Structure Definition:

```
struct Hotel {
    char name[50];
    char address[100];
    int grade;
    float avg_charge;
    int rooms;
};
```

- **name** – Name of the hotel
- **address** – Address/location of the hotel
- **grade** – Hotel grade (e.g., 1 to 5 stars)
- **avg_charge** – Average room charge (in ₹ or $)
- **rooms** – Number of rooms available

Steps Involved:

1. Create an array of hotels.
2. Accept details for each hotel using a loop.
3. Ask user to enter a **grade** to filter hotels.
4. Sort the hotels in ascending order of average room charges using **Bubble Sort**.
5. Print only those hotels that match the given grade.

Structure for Hotel and Sort by Room Charge

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

struct Hotel {
    char name[50];
    char address[100];
    int grade;
    float avg_charge;
    int rooms;
};

void printHotels(struct Hotel h[], int n, int search_grade) {
    struct Hotel temp;

    // Sort by avg_charge (ascending)
    for(int i = 0; i < n-1; i++) {
        for(int j = 0; j < n-i-1; j++) {
            if(h[j].avg_charge > h[j+1].avg_charge) {
                temp = h[j];
                h[j] = h[j+1];
                h[j+1] = temp;
            }
        }
    }

    printf("\nHotels with Grade %d:\n", search_grade);
    for(int i = 0; i < n; i++) {
        if(h[i].grade == search_grade) {
            printf("Name: %s, Address: %s, Charge: %.2f, Rooms: %d\n",
                h[i].name, h[i].address, h[i].avg_charge,
h[i].rooms);
        }
    }
}

int main() {
    struct Hotel h[100];
```

```
    int n, grade;

    printf("Enter number of hotels: ");
    scanf("%d", &n);

    for(int i = 0; i < n; i++) {
        printf("Hotel %d details:\n", i + 1);
        printf("Name: ");
        scanf("%s", h[i].name);
        printf("Address: ");
        scanf("%s", h[i].address);
        printf("Grade: ");
        scanf("%d", &h[i].grade);
        printf("Average Room Charge: ");
        scanf("%f", &h[i].avg_charge);
        printf("Number of Rooms: ");
        scanf("%d", &h[i].rooms);
    }

    printf("Enter grade to filter: ");
    scanf("%d", &grade);

    printHotels(h, n, grade);

    return 0;
}
```

B. Print Hotels with Charge Below a Limit

Objective:

- Display hotel details where the average room charge is **less than a user-specified value**.

Explanation:

- This involves filtering hotels based on a **maximum charge limit**.
- We loop through all hotel records and check if `avg_charge < max_value`.
- If yes, print the hotel details.

Why This Is Important:

- Helps customers or users **choose budget-friendly hotels**.
- Can be used in travel apps, booking systems, or hotel comparison tools.

Concepts Covered in Q14:

- Structures in C

- Arrays of structures
- Sorting (Bubble Sort)
- Conditional filtering
- User-defined functions
- Input/Output formatting

b. Filter Hotels with Room Charge Less Than a Given Value

```
void printCheapHotels(struct Hotel h[], int n, float max_charge) {
    printf("\nHotels with average charge less than %.2f:\n",
max_charge);
    for(int i = 0; i < n; i++) {
        if(h[i].avg_charge < max_charge) {
            printf("Name: %s, Address: %s, Grade: %d, Charge: %.2f,
Rooms: %d\n",
                   h[i].name, h[i].address, h[i].grade,
h[i].avg_charge, h[i].rooms);
        }
    }
}
```

Add a call in `main()` like:

```
float charge_limit;
printf("Enter maximum average charge to filter: ");
scanf("%f", &charge_limit);
printCheapHotels(h, n, charge_limit);
```
