2024

Xtreme Ends of PPS : Repeated PYQs

AIR

ACADEMIC INTERCONNECTED RESOURCES

Saumy

1/5/2024

Table of Contents

ACKNOWLEDGMENTS

I would like to thank Tanishttha for her insightful view of adding more PYQs instead of only the Repeated ones as "Agar aap sab bas unhe hi padhkar gaye to fail ho jaoge".

So , after that I added one more section of "Additional PYQs" on behalf of her insights . So, after the addition of this new section the probable chances of holding a Re on your shoulders become less but to make it more less I also added QR code section in them as if in case they by chance come in the long section parts you must know the basic insights of that topic .

I would also like to thanks my brother Nikhil who suggested me to make the document less formatted and colorful. So , I changed the fancy fonts to the more aesthetic ones like "Garamond" & "Times New Roman".

If there is any technical mistake in the document then you can correct it from your end as the document is now released.

Question 1

Conditional Statements

Write a program to find

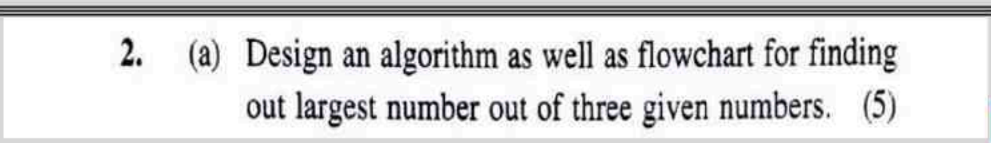
Greatest Among 3 Numbers

2019 (5 Marks) [Ques. { 2(a) }]

March 2023 (1.5 Marks) {Ques. 1(j) }

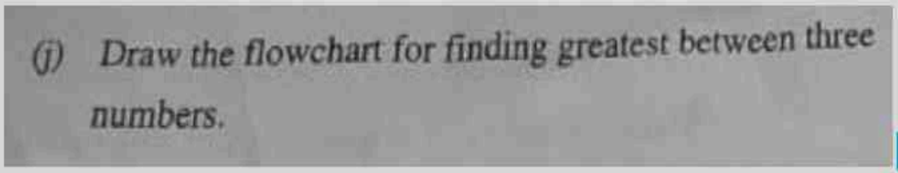
December 2023 (10 Marks) [Ques. { 2(a) }]

TOPIC : Conditional Statements



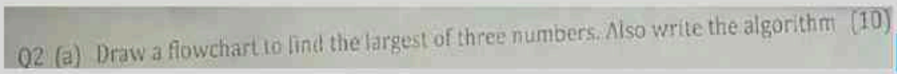
2. (a) Design an algorithm as well as flowchart for finding out largest number out of three given numbers. (5)

[2019]



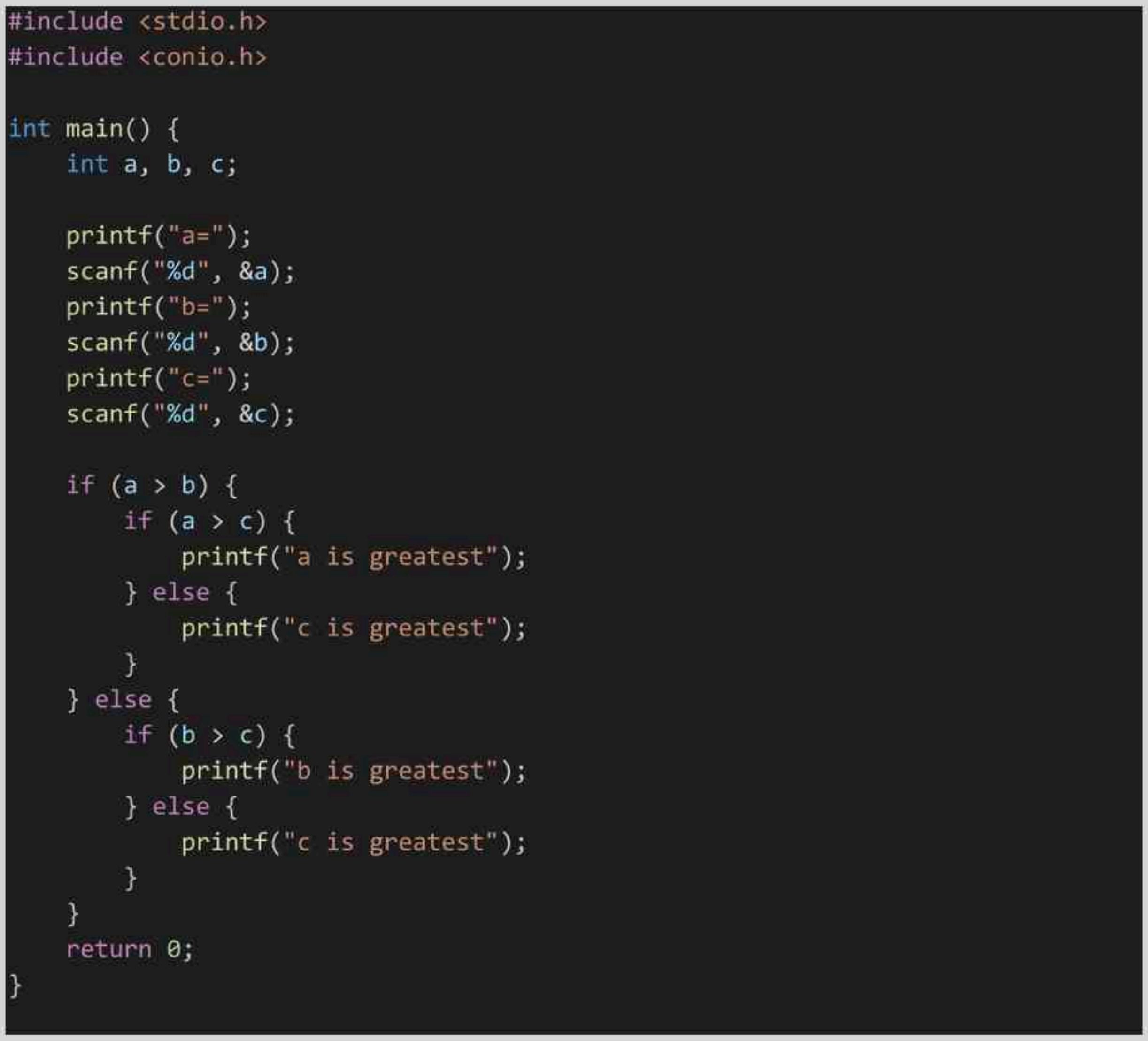
(j) Draw the flowchart for finding greatest between three numbers.

[MARCH 2023]

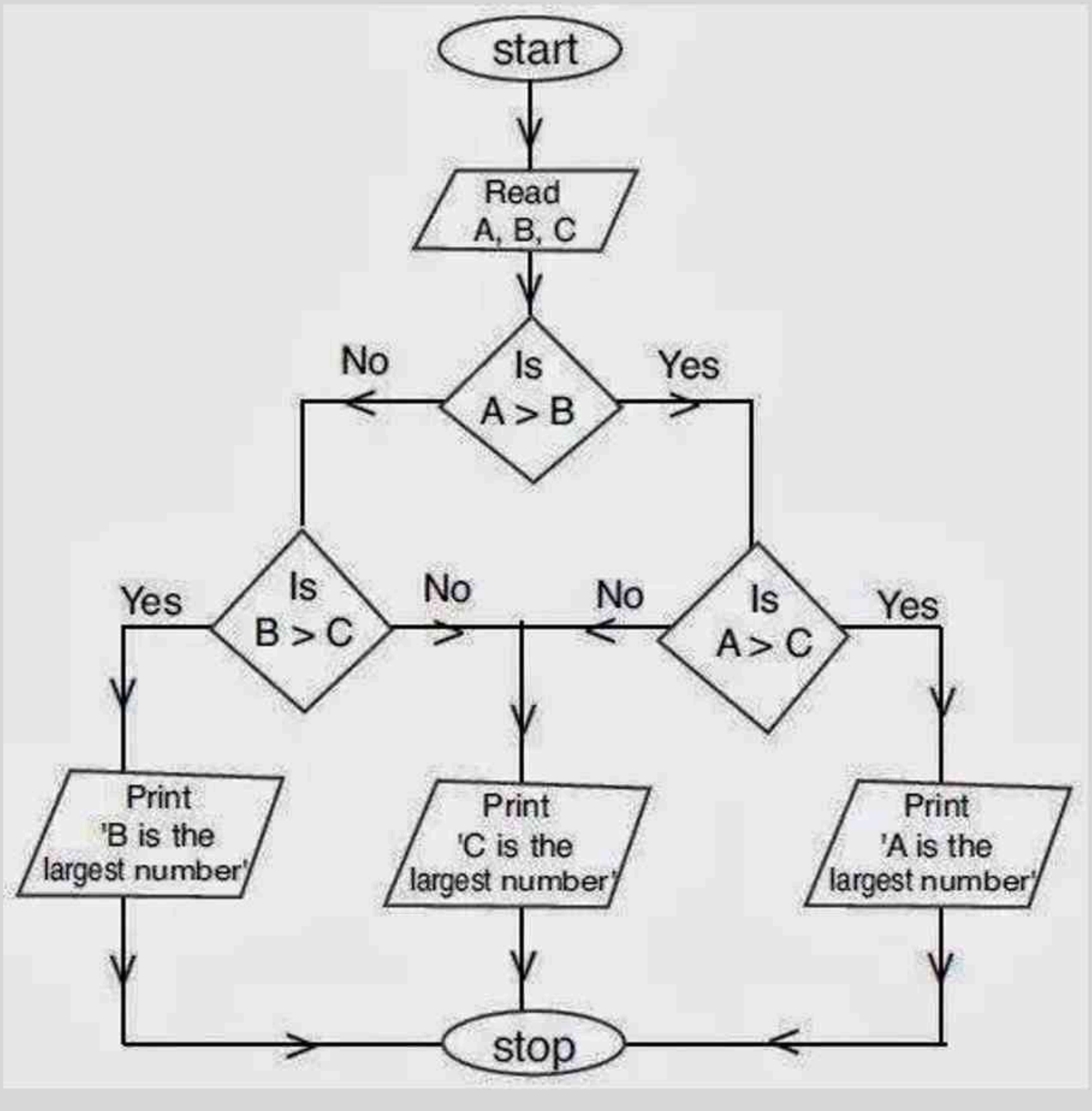


Q2 (a) Draw a flowchart to find the largest of three numbers. Also write the algorithm (10)

[DEC 2023]



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

int main() {
    int a, b, c;

    printf("a=");
    scanf("%d", &a);
    printf("b=");
    scanf("%d", &b);
    printf("c=");
    scanf("%d", &c);

    if (a > b) {
        if (a > c) {
            printf("a is greatest");
        } else {
            printf("c is greatest");
        }
    } else {
        if (b > c) {
            printf("b is greatest");
        } else {
            printf("c is greatest");
        }
    }
    return 0;
}
```


start
Read A, B, C
Is A > B
No
Yes
Is B > C
Yes
No
Is A > C
No
Yes
Print 'B is the largest number'
Print 'C is the largest number'
Print 'A is the largest number'
stop

Note: You can write the ALGORITHM simply by describing & explaining the above given code & the flowchart

Question 2

Functions

Write a program to

Swap 2 Numbers

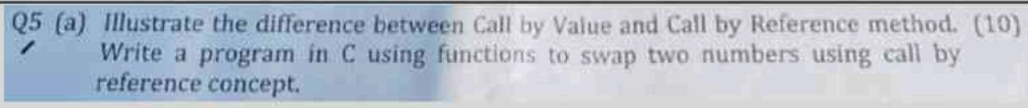
(Call By reference/Call by value)

2022 (10 Marks) [Ques. { 5(a) }]

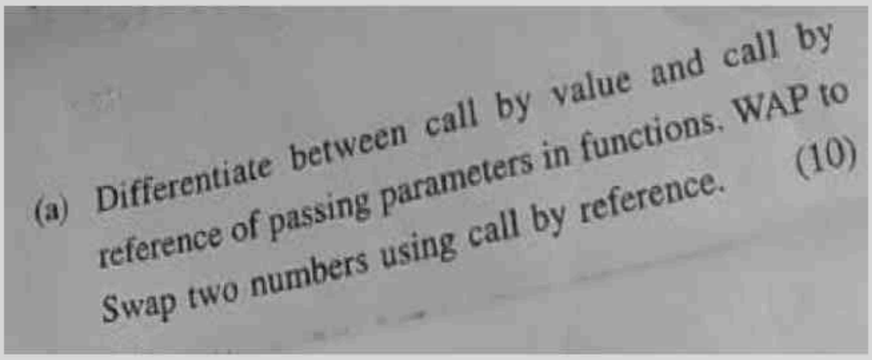
March 2023 (10 Marks) [Ques. { 5(a) }]

December 2023 (10 Marks) [Ques. { 5(b) }]

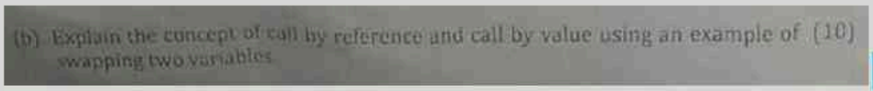
TOPIC : Functions


Q5 (a) Illustrate the difference between Call by Value and Call by Reference method. (10)
Write a program in C using functions to swap two numbers using call by reference concept.

[2022]


(a) Differentiate between call by value and call by reference of passing parameters in functions. WAP to Swap two numbers using call by reference. (10)

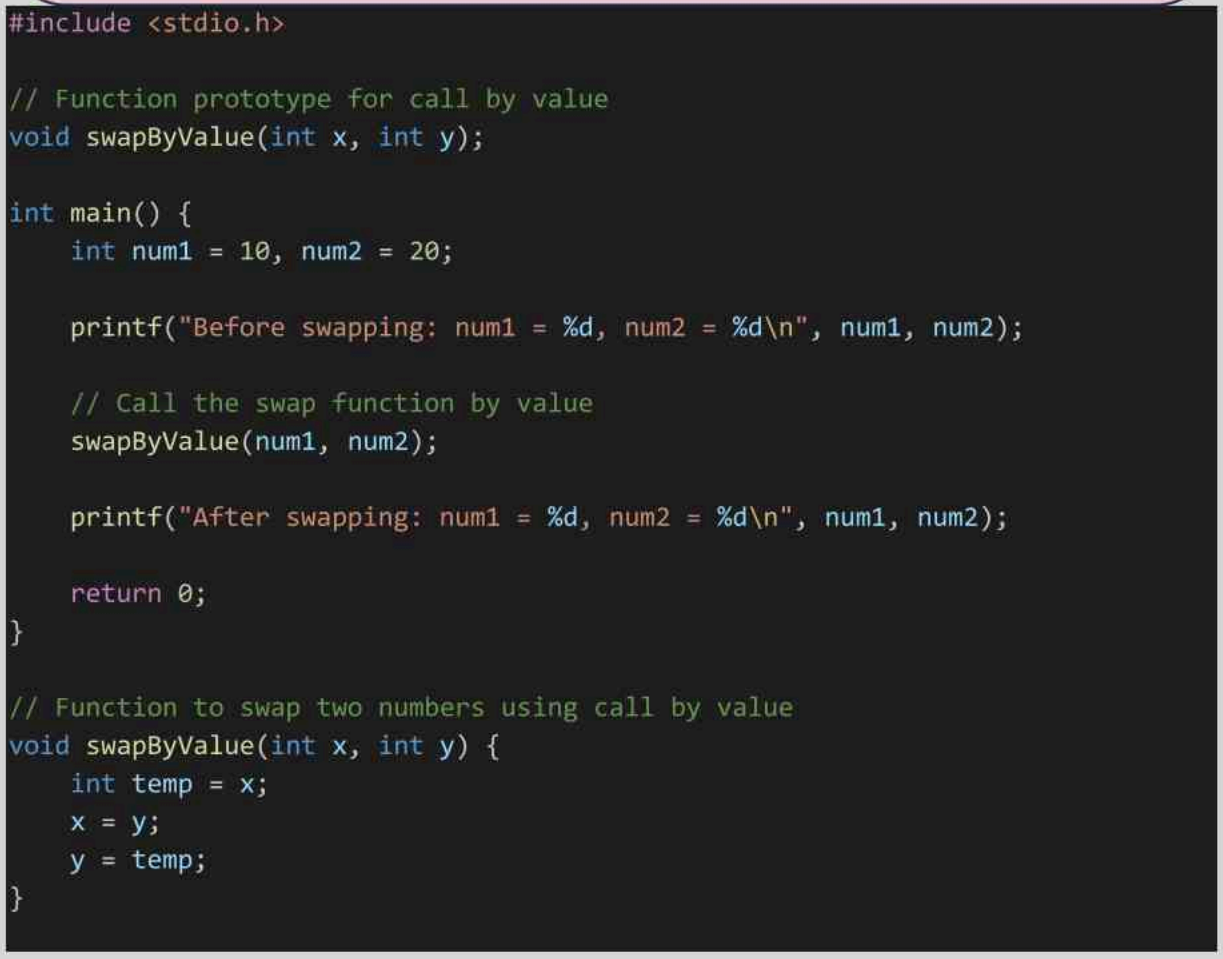
[MARCH 2023]


(b) Explain the concept of call by reference and call by value using an example of swapping two variables (10)

[DEC 2023]

Call by Value :

In call by value, the values of the actual parameters (arguments) are copied to the formal parameters of the function. Changes made to the formal parameters inside the function have no effect on the actual parameters outside the function.



```c
#include <stdio.h>

// Function prototype for call by value
void swapByValue(int x, int y);

int main() {
    int num1 = 10, num2 = 20;

    printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

    // Call the swap function by value
    swapByValue(num1, num2);

    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

    return 0;
}

// Function to swap two numbers using call by value
void swapByValue(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
```

Call by Reference :

In call by reference, the addresses of the actual parameters are passed to the function, allowing the function to directly modify the values of the variables outside the function.

```c
#include <stdio.h>

// Function prototype for call by reference
void swapByReference(int *x, int *y);

int main() {
    int num1 = 10, num2 = 20;

    printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

    // Call the swap function by reference
    swapByReference(&num1, &num2);

    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

    return 0;
}

// Function to swap two numbers using call by reference
void swapByReference(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}
```

Question 3

Recursion

Write a program to

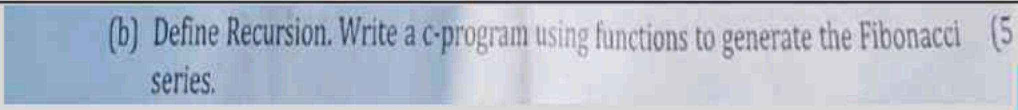
Generate Fibonacci Series

(Using Recursion/Function)

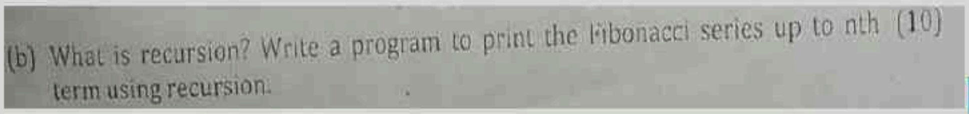
2022 (5 Marks) [Ques. { 5(b) }]

December 2023 (10 Marks) [Ques. { 3(b) }]

TOPIC : Recursion


(b) Define Recursion. Write a c-program using functions to generate the Fibonacci series. (5

[2022]


(b) What is recursion? Write a program to print the Fibonacci series up to nth term using recursion. (10)

[DEC 2023]

Recursion Basic Code

```
if
(
    ---------
    ---------
    Base Case
    ---------
    ---------
)
else
(
    ---------
    ---------
    Recursive case
    ---------
    ---------
)
```

To understand Recursion , we'll take an example to :

Printing Name 'n' Times Using Recursion

```
#include<stdio.h>

int main()
{
int n;
int fun(int);
printf("n=");
scanf("%d",&n);
fun(n);
}
int fun(int n)
{
    if (n==1)
        {
            printf("Saumy");
        }
    else
        {
        printf("Saumy\n");
        fun(n-1);
        }
return 0;
}
```

Fibonacci Series Code

```
#include <stdio.h>

// Function prototype
int CalFib(int term);

int main() {
    int n, i;

    // Input the number of terms
    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci series up to %d terms:\n", n);
    for (i = 0; i < n; i++) {
        printf("%d ", CalFib(i));
    }

    return 0;
}

// Recursive function to calculate nth Fibonacci number
int CalFib(int term) {
    if (term <= 1)
        return term;
    else
        return CalFib(term - 1) + CalFib(term - 2);
}
```

Question 4

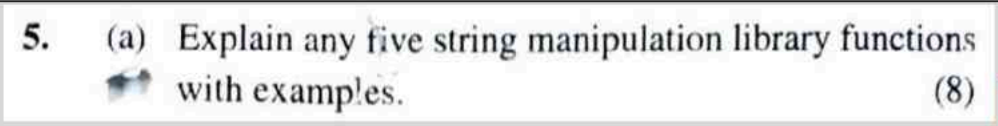
Strings

String

2019 (8 Marks) [Ques. { 5(a) }]

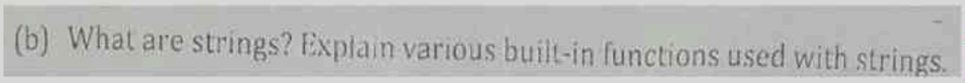
December 2023 (5 Marks) [Ques. { 6(b) }]

TOPIC : Strings



5. (a) Explain any five string manipulation library functions with examples. (8)

[2019]



(b) What are strings? Explain various built-in functions used with strings.

[DEC 2023]

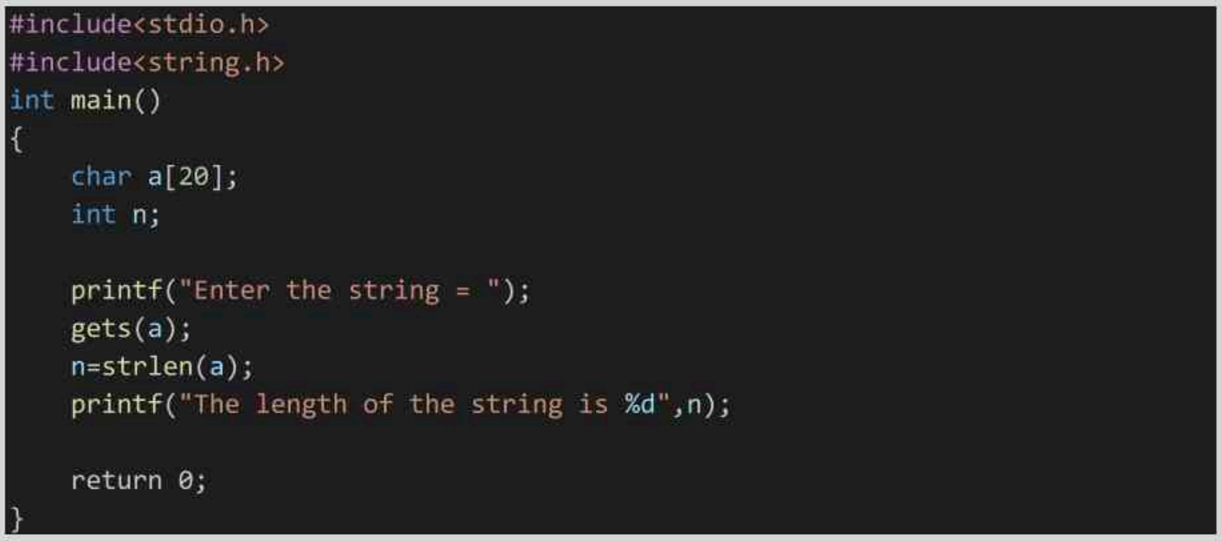
- String is a sequence of character that is treated as a single data item & terminated by '\0' .
- ASCII value of null character is 0 .
- We use gets(string name) & puts(string name)

String function

- strlen()
- strrev()
- strlwr()
- strupr()
- strcpy()

➢ strlen()

To calculate the length (no. of character)

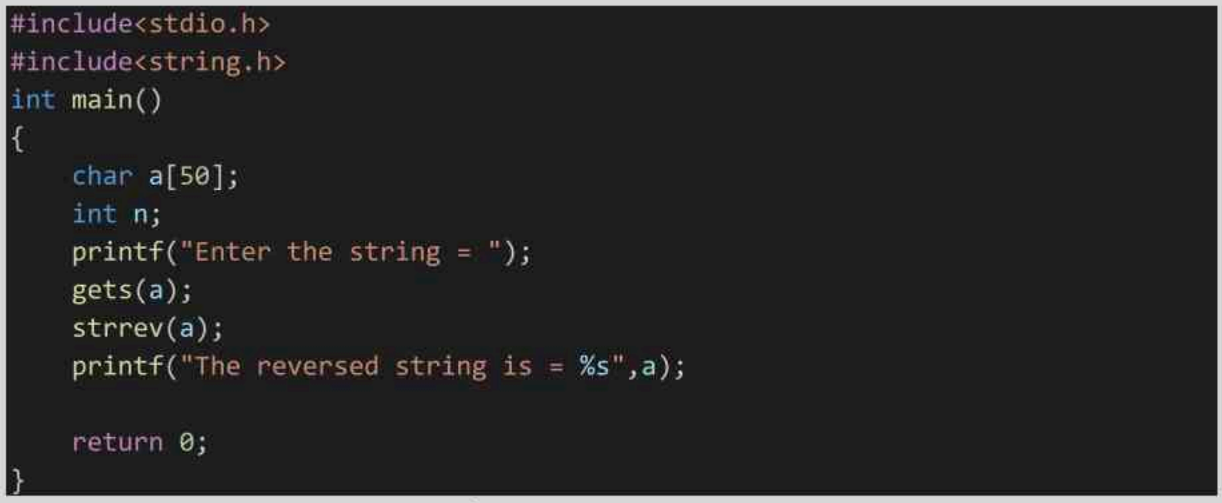


```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[20];
    int n;

    printf("Enter the string = ");
    gets(a);
    n=strlen(a);
    printf("The length of the string is %d",n);

    return 0;
}
```

➢ strrev()

To reverse the string

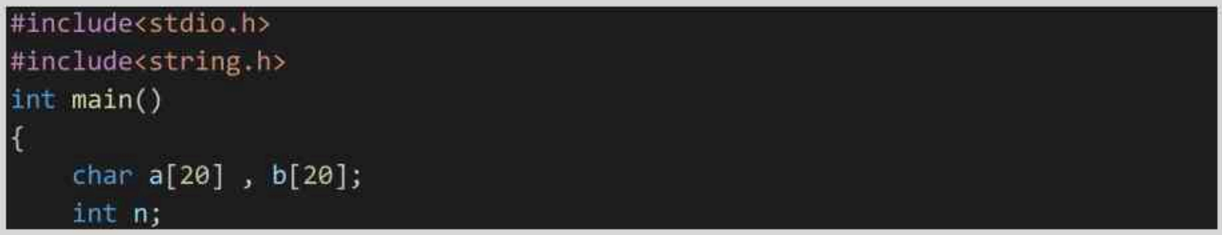
Like : Keep ➔ Peek



```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[50];
    int n;
    printf("Enter the string = ");
    gets(a);
    strrev(a);
    printf("The reversed string is = %s",a);

    return 0;
}
```

➢ strcat()

To join 2 strings



```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[20] , b[20];
    int n;
    printf("1st string = ");
    gets(a);
    printf("2nd string = ");
    gets(b);
    strcat(a,b);
    printf("join string = %s",a);

    return 0;
}
```

➢ strcpy()

Used to copy particular string in to another string

```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[50],b[50];

    printf("1st string = ");
    gets(a);
    strcpy(b,a);
    printf("2nd string = ");
    puts(b);

    return  0;
}
```

➢ strupr()

```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[50];

    printf("Enter the string = ");
    gets(a);
    printf("%s", strupr(a));

    return 0;
}
```

Question 5

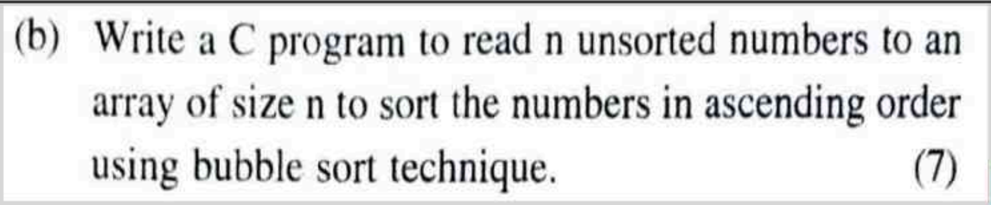
Bubble Sort

Bubble Sort

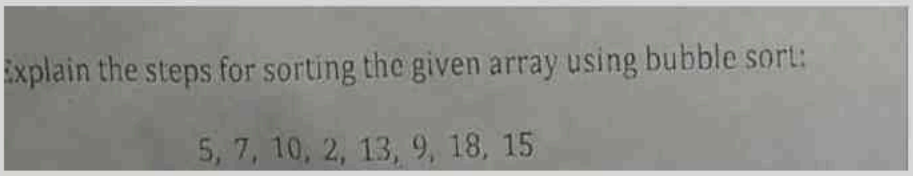
2019 (7 Marks) [Ques. { 5(b) }]

December 2023 (5 Marks) [Ques. { 5(a) }]

TOPIC : Sorting


(b) Write a C program to read n unsorted numbers to an array of size n to sort the numbers in ascending order using bubble sort technique. (7)

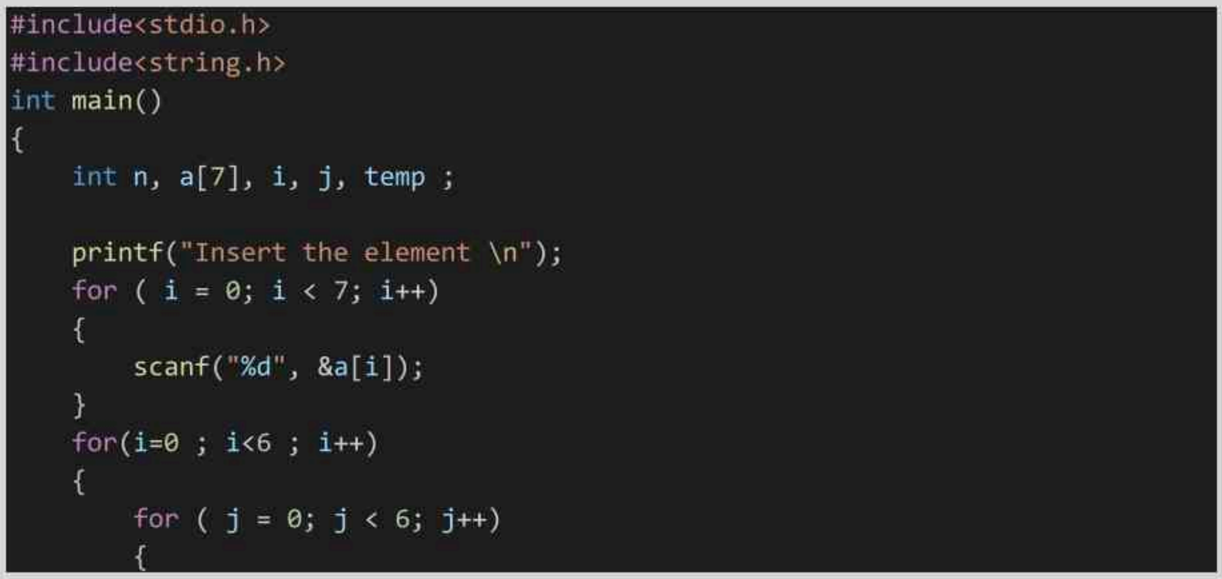
[2019]


Explain the steps for sorting the given array using bubble sort:

5, 7, 10, 2, 13, 9, 18, 15

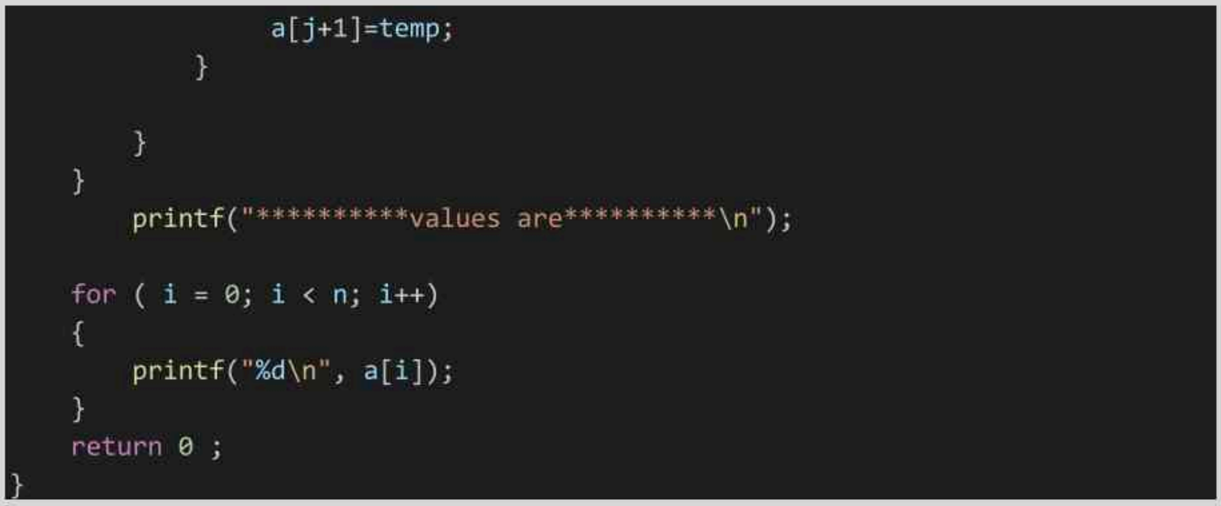
[DEC 2023]

For a Basic 7 elements array as an example


```
#include<stdio.h>
#include<string.h>
int main()
{
    int n, a[7], i, j, temp ;

    printf("Insert the element \n");
    for ( i = 0; i < 7; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0 ; i<6 ; i++)
    {
        for ( j = 0; j < 6; j++)
        {
            if( a[j] < a[j+1]) //Reverse the operator sign to sort the provided
array in ascending order
            {
                temp = a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }

        }
    }
        printf("**********values are**********\n");

    for ( i = 0; i < 7; i++)
    {
        printf("%d\n", a[i]);
    }
    return 0 ;
}
```

Sorting using Bubble Sort for 'n' size array

```
#include<stdio.h>
#include<string.h>
int main()
{
    int n, a[10000], i, j, temp ;

    printf("n=");
    scanf("%d",&n);
    printf("Insert the element \n");
    for ( i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0 ; i<n-1 ; i++)
    {
        for ( j = 0; j < n-1; j++)
        {
            if( a[j] < a[j+1]) //Reverse the operator sign to sort the provided
array in ascending order
            {
                temp = a[j];
                a[j]=a[j+1];
```



```
                a[j+1]=temp;
            }

        }
    }
        printf("**********values are**********\n");

    for ( i = 0; i < n; i++)
    {
        printf("%d\n", a[i]);
    }
    return 0 ;
}
```

Question 6

Differentiate Between

Compiler & Interpreter

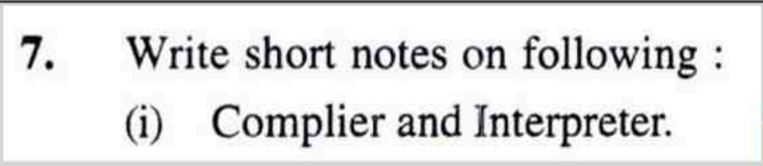
while & do-while loop

2019(i) (5 Marks) [Ques. { 7(i) }]

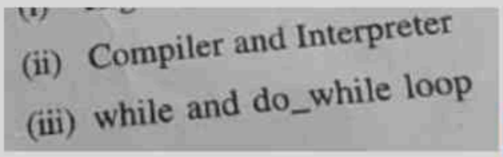
March 2023(i & ii) (10 Marks) [Ques. {7(i & ii)}]

December 2023(i) (5 Marks) [Ques. { 7(a) }]

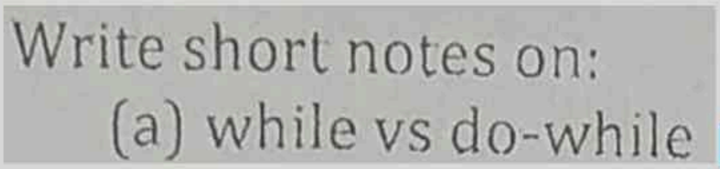
TOPIC : Theoretical


7. Write short notes on following :
(i) Complier and Interpreter.

[2019]


(ii) Compiler and Interpreter
(iii) while and do_while loop

[MARCH 2023]


Write short notes on:
(a) while vs do-while

[DEC 2023]

Feature	Compiler	Interpreter
Processing	Converts entire program into machine code before execution	Translates code line-by-line during execution
Output	Generates intermediate object code or executable file	No separate object code generation, directly executes code

Execution Speed	Generally faster as code is precompiled	Generally slower as code is translated during runtime
Error Handling	Detects all errors before execution, requires full compilation to detect error	Stops at first encountered error, easier to identify source of error
Memory Usage	Typically requires more memory during compilation	Requires less memory during execution as code is not precompiled
Portability	Executable file can be run on any compatible system without need for source code	Requires interpreter installation on each system where code needs to run
Debugging	Debugging may be harder as source code may be different from machine cod	Easier to debug as interpreter can provide detailed error message
Modification	Any changes require recompilation of entire program	Changes can be made quickly without need for recompilation

Examples	C, C++, Java	Python, Ruby, JavaScript
Usage	Used for performance-critical applications or where code will be executed multiple time	Often used in scripting languages or for rapid development/testing

Feature	while Loop	do-while Loop
Syntax	while (condition) { statement(s) };	do { statement(s); } while (condition);
Execution	Condition is evaluated before executing the loop body	Condition is evaluated after executing the loop body
Entry Control	May not execute loop body if condition is initially false	Always executes the loop body at least once
Exit Control	May not execute loop body if condition is initially false	Executes the loop body at least once, condition checked after loop body

Use Case	Suitable when loop body may not need to be executed at all	Suitable when loop body must be executed at least once
Loop Termination	Depends on the condition evaluation	Depends on the condition evaluation
Iteration Control	Determined solely by the condition evaluation	Condition determines whether to continue or terminate loop
Error-Prone	May result in infinite loop if condition is never met	Less prone to infinite loops as loop body always executes at least once

Question 7

Operating System

Operating System

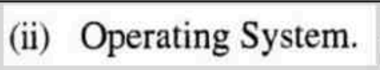
2019 (5 Marks) [Ques. { 7(ii) }]

2022 (1.5 Marks) [Ques. { 1(j) }]

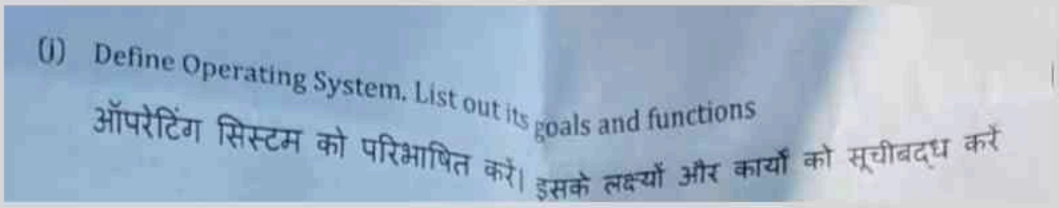
March 2023 (10 Marks) {Ques. 3(a) }

December 2023 (5 Marks) [Ques. { 2(b) }]

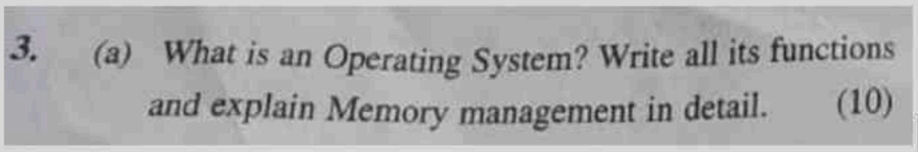
TOPIC : **Theoretical**


(ii) Operating System.

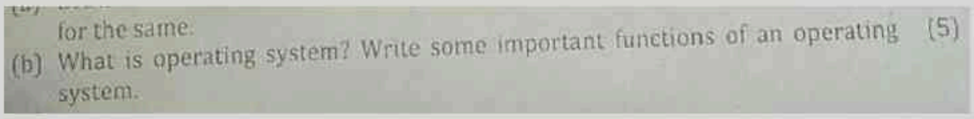
[2019]


(i) Define Operating System. List out its goals and functions
ऑपरेटिंग सिस्टम को परिभाषित करें। इसके लक्ष्यों और कार्यों को सूचीबद्ध करें

[2022]


3. (a) What is an Operating System? Write all its functions and explain Memory management in detail. (10)

[March 2023]


for the same.
(b) What is operating system? Write some important functions of an operating system. (5)

[Dec 2023]

Operating System Definition

An operating system (OS) is a software that acts as an intermediary between computer hardware and user applications. It manages hardware resources and provides essential services to software applications.

Goals of an Operating System

Efficiency	Ease of Use	Maintainability
Efficiently utilizes hardware resources to maximize system performance and responsiveness.	Provides a user-friendly interface for interacting with the system and applications.	Facilitates easy maintenance and updates by allowing seamless installation, configuration, and troubleshooting.

Functions of an Operating System

Process Management	**Security and Access Control**	**User Interface**
• **Creates, schedules, and terminates processes.**	• Enforces user authentication and access control mechanisms.	• Provides interfaces for user interaction, including command-line interfaces (CLI) and graphical user interfaces (GUI).
• **Manages process synchronization and communication.**	• Implements encryption, firewalls, and intrusion detection	• Supports input/output devices such as

• Allocates and deallocates process resources.	systems to protect system integrity and confidentiality.	keyboards, mice, and touchscreens.

Question 8

Structures

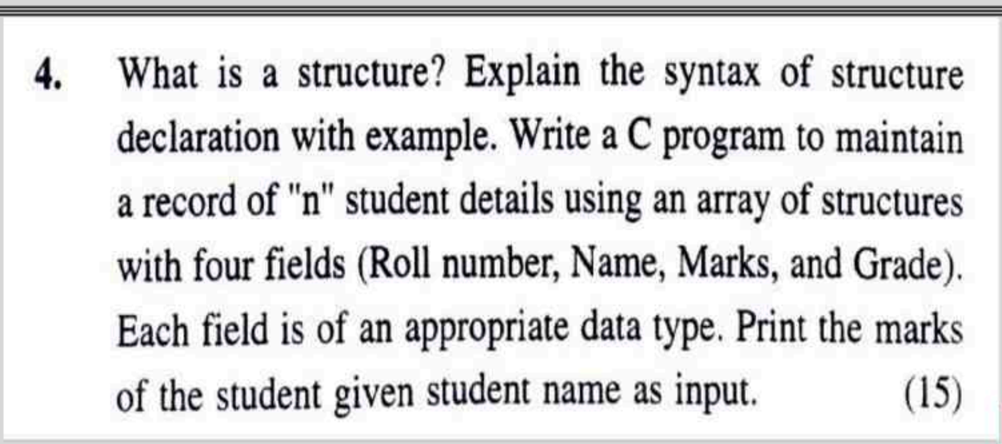
Structures

2019 (15 Marks) [Ques. 4]

2022 (10 Marks) [Ques. { 6(a) }]

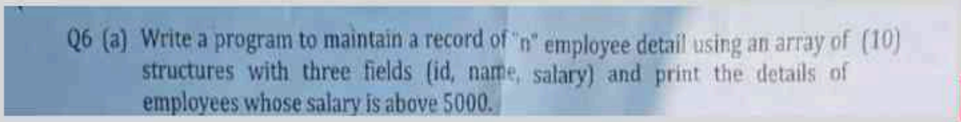
December 2023 (15 Marks) [Ques. 4]

TOPIC : Structures



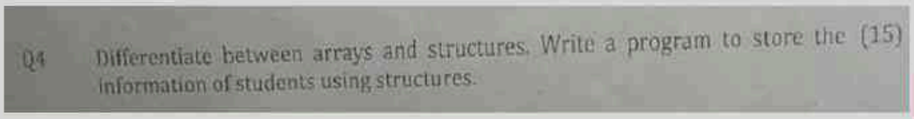
4. What is a structure? Explain the syntax of structure declaration with example. Write a C program to maintain a record of "n" student details using an array of structures with four fields (Roll number, Name, Marks, and Grade). Each field is of an appropriate data type. Print the marks of the student given student name as input. (15)

[2019]



Q6 (a) Write a program to maintain a record of "n" employee detail using an array of structures with three fields (id, name, salary) and print the details of employees whose salary is above 5000. (10)

[2022]



Q4 Differentiate between arrays and structures. Write a program to store the information of students using structures. (15)

[DEC 2023]

Structures

A collection of values of different data types .

Example :

For a student store the following :

name (String)

roll no (Integer)

cgpa (Float)

DEFINING DATA TYPE

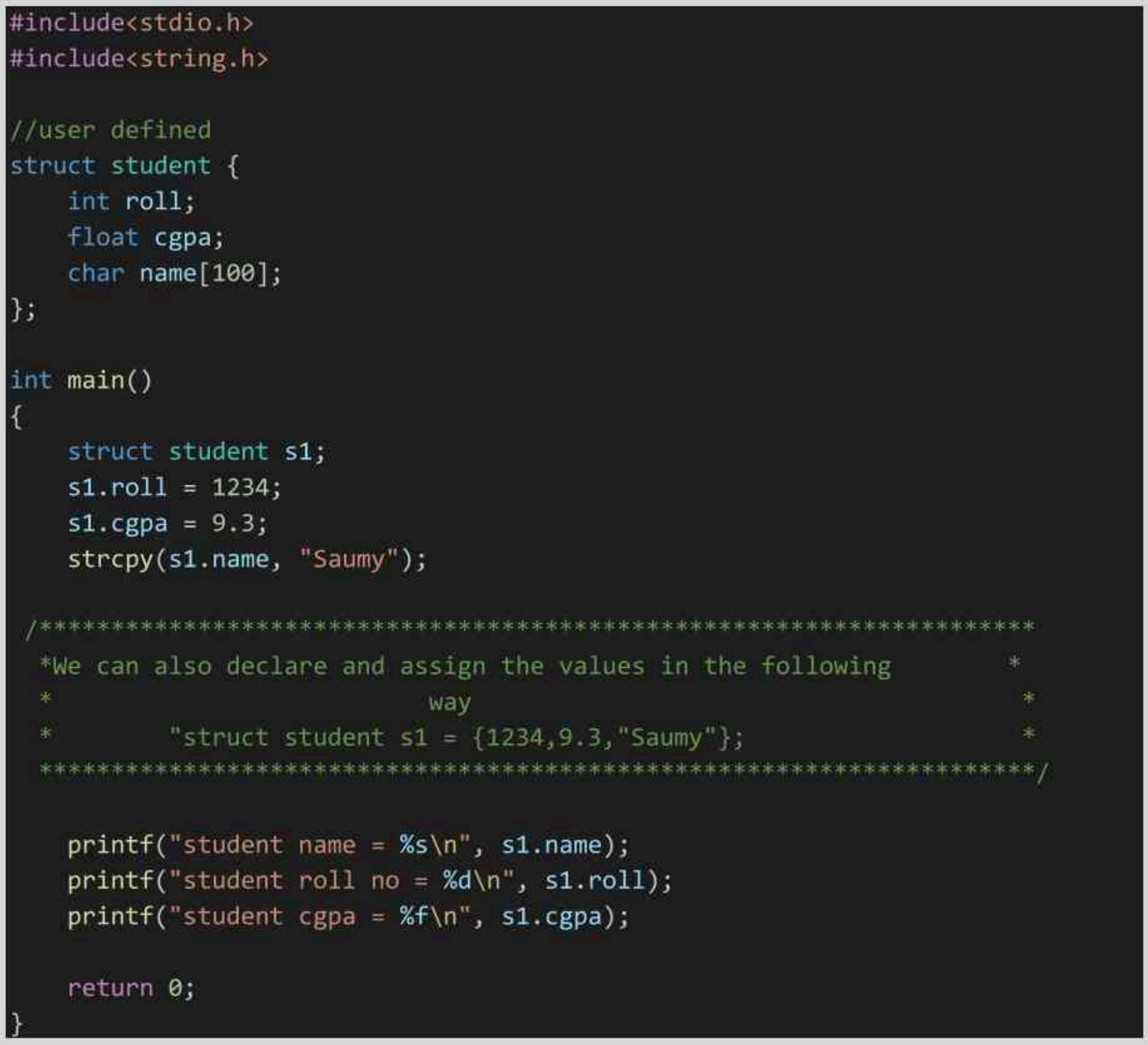
Syntax

```
struct student {
char name [100];
int roll;
float cgpa;
}.
```

Declaring & Using Data Type

```
struct student s1;

s1.cgpa = 9.3
```

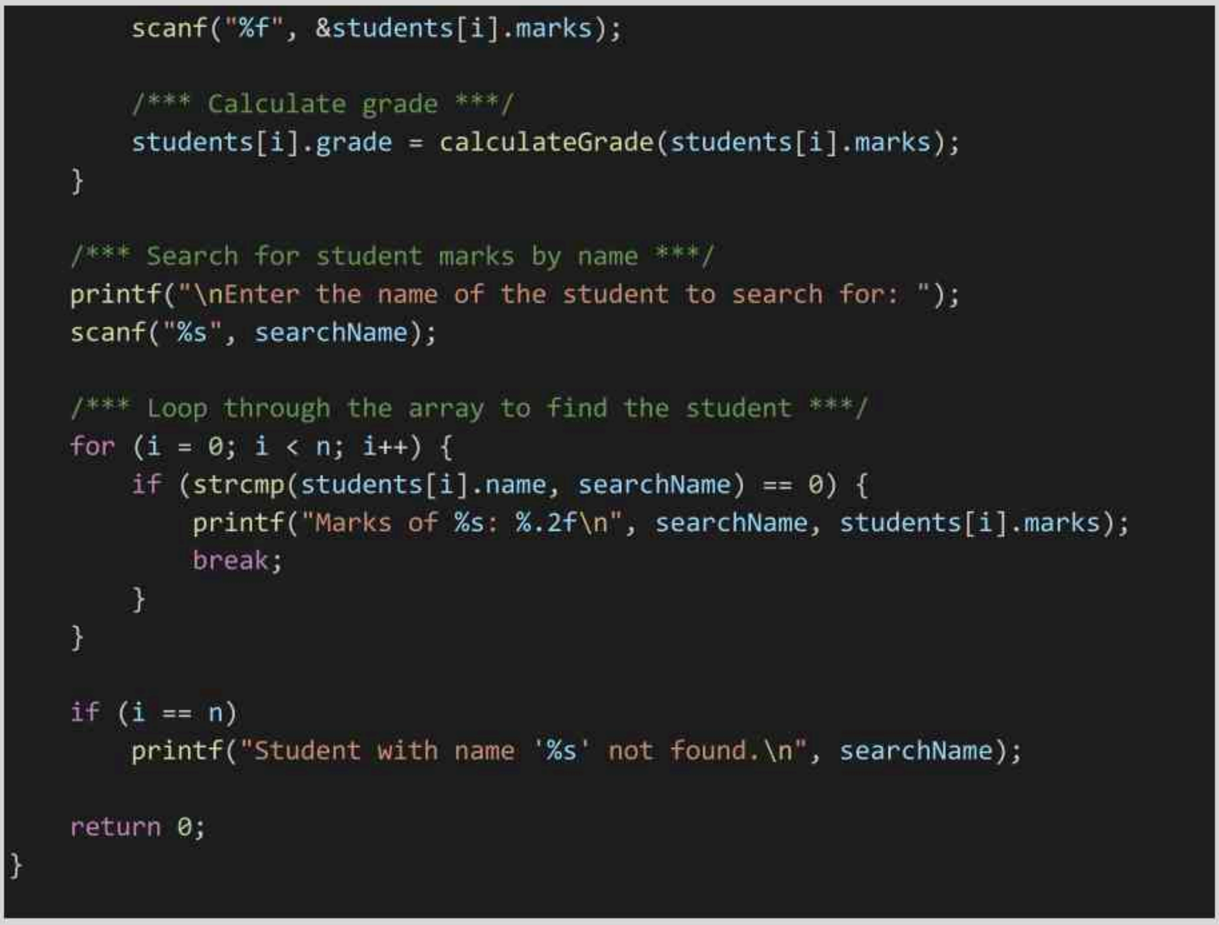
Basic Example of a Structure



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

//user defined
struct student {
    int roll;
    float cgpa;
    char name[100];
};

int main()
{
    struct student s1;
    s1.roll = 1234;
    s1.cgpa = 9.3;
    strcpy(s1.name, "Saumy");

 /*********************************************************************
  *We can also declare and assign the values in the following        *
  *                           way                                     *
  *         "struct student s1 = {1234,9.3,"Saumy"};                  *
  *********************************************************************/

    printf("student name = %s\n", s1.name);
    printf("student roll no = %d\n", s1.roll);
    printf("student cgpa = %f\n", s1.cgpa);

    return 0;
}
```

Program to maintain record of 'n' students

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

#define MAX_STUDENTS 50
#define MAX_NAME_LENGTH 50
/***I used #define to define constants because it's a common way to declare
constants in C programs***/

/*** Structure to store student details ***/
struct Student {
    int rollNumber;
    char name[MAX_NAME_LENGTH];
    float marks;
    char grade;
};

/*** Function to calculate grade based on marks ***/
char calculateGrade(float marks) {
    if (marks >= 90)
        return 'A';
    else if (marks >= 80)
        return 'B';
    else if (marks >= 70)
        return 'C';
    else if (marks >= 60)
        return 'D';
    else
        return 'F';
}

int main() {
    struct Student students[MAX_STUDENTS];
    int n, i;
    char searchName[MAX_NAME_LENGTH];

    /*** Input the number of students ***/
    printf("Enter the number of students: ");
    scanf("%d", &n);

    /*** Input student details ***/
    for (i = 0; i < n; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Roll Number: ");
        scanf("%d", &students[i].rollNumber);
        printf("Name: ");
        scanf("%s", students[i].name);
        printf("Marks: ");
```

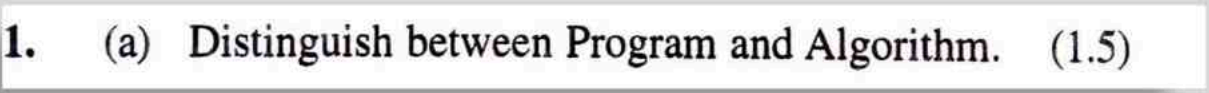


```
        scanf("%f", &students[i].marks);

        /*** Calculate grade ***/
        students[i].grade = calculateGrade(students[i].marks);
    }

    /*** Search for student marks by name ***/
    printf("\nEnter the name of the student to search for: ");
    scanf("%s", searchName);

    /*** Loop through the array to find the student ***/
    for (i = 0; i < n; i++) {
        if (strcmp(students[i].name, searchName) == 0) {
            printf("Marks of %s: %.2f\n", searchName, students[i].marks);
            break;
        }
    }

    if (i == n)
        printf("Student with name '%s' not found.\n", searchName);

    return 0;
}
```

Additional Short PYQs

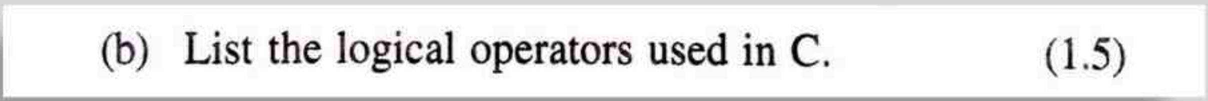
2019 Short PYQs

Short 1 ➔


1. (a) Distinguish between Program and Algorithm. (1.5)

Algorithm	**Program**
1. Used in Design time.	1. Used in implementation time
2. We can use any language (English or Mathematical notation)	2. We can use only Programming language (like C , C++ , Java etc.)
3. Algorithms are independent of Hardware/Software/ OS.	3. Programs are independent of Hardware/Software/ OS.
4. After writing an Algorithm , we do Analysis .	4. But in case of Programs , we do Testing.

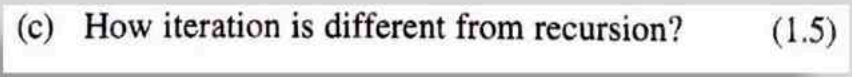
Short 2 ➔



(b) List the logical operators used in C. (1.5)

Logical AND (&&)	Logical OR (||)	Logical NOT (!)
Returns true if both operands are true.	Returns true if either of the operands is true.	Returns true if the operand is false, and vice versa. It's a unary operator.

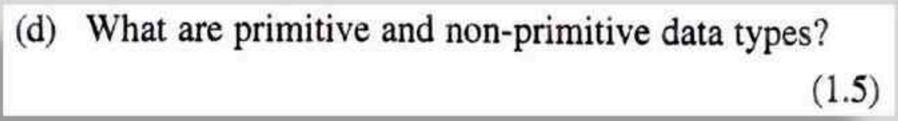
Short 3 ➔



(c) How iteration is different from recursion? (1.5)

Iteration	Recursion
1. It is a process of executing certain set of instructions repeatedly , without calling the self function.	1. It is a process of executing certain set of instructions repeatedly by calling the self function repeatedly.
2. Memory utilization is less.	2. More memory utilization.
3. Simple to implement.	3. Complex implementation.
4. More line of code	4. Compactness

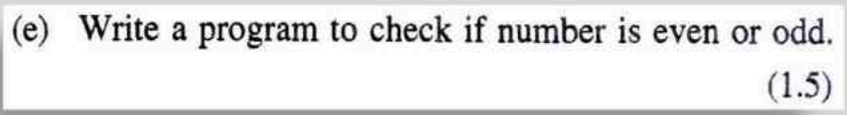
Short 4 ➔


(d) What are primitive and non-primitive data types?

(1.5)

Category	Description	Examples
Primitive Data Types	Fundamental data types directly supported by the programming language.	Integers, floating-point numbers, characters, Booleans.
Non-Primitive Data Types	Derived data types that are composed of primitive data types or other non-primitive data types.	Arrays, structures, pointers, classes, interfaces.

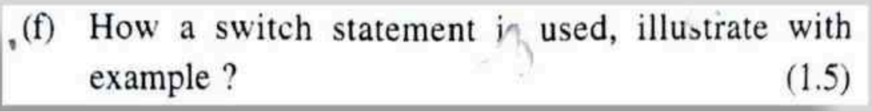
Short 5 ➔


(e) Write a program to check if number is even or odd.

(1.5)

```c
#include <stdio.h>

int main() {
    int num;

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

    if (num % 2 == 0) {
        printf("%d is even.\n", num);
    } else {
        printf("%d is odd.\n", num);
    }

    return 0;
}
```

Short 6 ➔


(f) How a switch statement is used, illustrate with example ? (1.5)

A switch statement in C is used to select one of several code blocks to be executed, depending on the value of a variable or an expression.

```c
#include <stdio.h>

int main() {
    int choice;

    // Prompt the user for a choice
    printf("Enter a number between 1 and 3: ");
    scanf("%d", &choice);

    // Switch statement to execute different code blocks based on the value of
'choice'
    switch (choice) {
        case 1:
            printf("You chose option 1.\n");
            break;
        case 2:
            printf("You chose option 2.\n");
            break;
        case 3:
            printf("You chose option 3.\n");
            break;
        default:
            printf("Invalid choice. Please enter a number between 1 and 
3.\n");
    }

    return 0;
}
```

In this example, the user is prompted to enter a number between 1 and 3. The entered value is stored in the variable **choice**. The switch statement then evaluates the value of **choice** and executes the corresponding code block associated with each case label.

- If **choice** is 1, it prints "You chose option 1."
- If **choice** is 2, it prints "You chose option 2."
- If **choice** is 3, it prints "You chose option 3."
- If **choice** doesn't match any of the case labels (i.e., it's not 1, 2, or 3), it executes the code block associated with the **default** case, printing "Invalid choice. Please enter a number between 1 and 3."

Each case label is followed by a colon (**:**) and the corresponding code block. The **break** statement is used to exit the switch statement once a matching case is found.

Short 7 ➔

(g) What is the use of getchar and putchar? (1.5)

getchar : It is used to input a single character

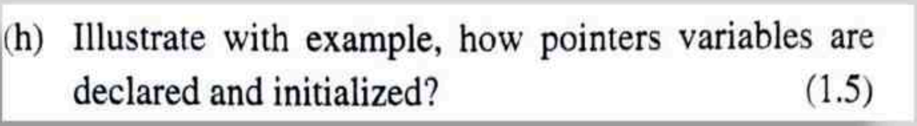
char **x**;

x=getchar();

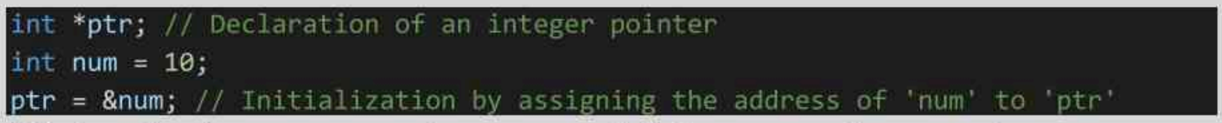
putchar : It is used top print a single character

putchar(x);

Short 8 ➔

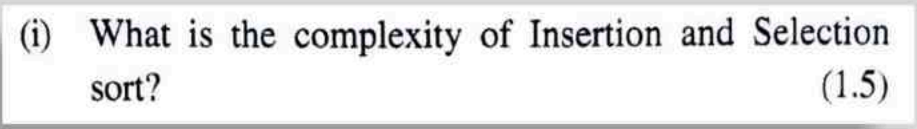

(h) Illustrate with example, how pointers variables are declared and initialized? (1.5)

Pointer variables in C are declared by specifying the data type they point to, followed by an asterisk (*), and then the variable name. Initialization involves assigning the memory address of another variable using the address-of operator (&) or assigning NULL for uninitialized pointers. For example:


```
int *ptr; // Declaration of an integer pointer
int num = 10;
ptr = &num; // Initialization by assigning the address of 'num' to 'ptr'
```

This declares a pointer variable **ptr** that points to an integer, and initializes it to the address of the integer variable **num**.

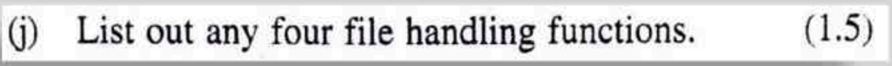
Short 9



(i) What is the complexity of Insertion and Selection sort? (1.5)

O(n^2)

Short 10 ➔

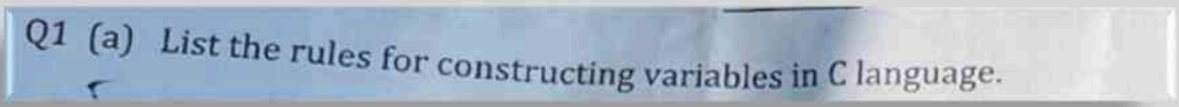


(j) List out any four file handling functions. (1.5)

1. **fopen()**: Used to open a file, providing options for reading, writing, or appending data.
2. **fclose()**: Closes a file that was opened using fopen(), ensuring that any buffered data is written to the file.
3. **fread()**: Reads data from a file into a buffer.
4. **fwrite()**: Writes data from a buffer to a file.

2022 Short PYQs

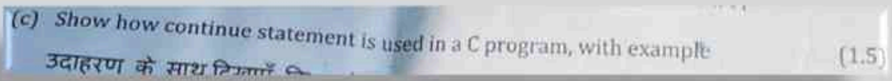
Short 1 ➔



Q1 (a) List the rules for constructing variables in C language.

1. Variable names can consist of letters, digits, and underscores.
2. The first character must be a letter or an underscore.
3. Variable names are case-sensitive.
4. Keywords (reserved words) cannot be used as variable names.

Short 2 ➔

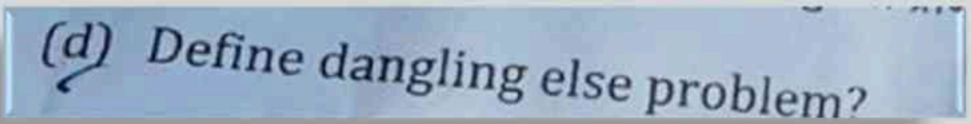

(c) Show how continue statement is used in a C program, with example (1.5)
उदाहरण के साथ दिखाएँ

The **continue** statement in C is used to skip the rest of the loop's body and jump to the next iteration. It's commonly used to avoid executing certain statements under specific conditions.

For example :

```
#include <stdio.h>

int main() {
    int i;
    for (i = 1; i <= 5; i++) {
        if (i == 3) {
            continue; // Skip iteration when i equals 3
        }
        printf("%d\n", i);
    }
    return 0;
}
```

Short 3


(d) Define dangling else problem?

The dangling else problem occurs in C with nested if statements when the "else" part isn't clearly associated with a specific "if" condition.

Imagine multiple if statements without curly braces:

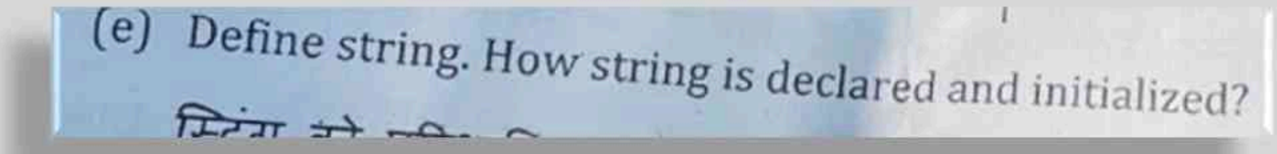
```
if (condition1)
  statement1;
```

```c
if (condition2)
  statement2;
else
  statement3; // Dangling else - unclear which if it belongs to
```

Here, the compiler might pair the else with the second "if" (statement2), leading to unintended behavior.

This ambiguity can cause bugs and unexpected results.

Short 4



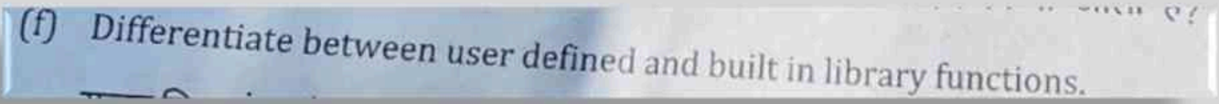
(e) Define string. How string is declared and initialized?

A string in C is an array of characters terminated by a null character ('\0'). It represents a sequence of characters, often used for storing and manipulating textual data. Strings are declared as character arrays with a specified size or as pointers to characters. Initialization can be done by assigning a string literal enclosed in double quotes or by assigning the address of a character array containing the string data.

For example :

```c
char str1[10] = "Hello"; // Declaration and initialization using array
char *str2 = "World"; // Declaration and initialization using pointer
```

Short 5 ➔



(f) Differentiate between user defined and built in library functions.

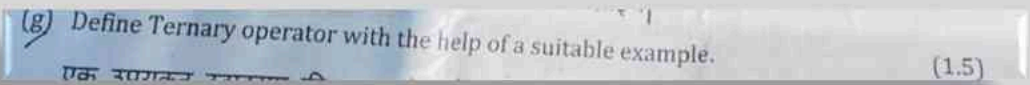
User-Defined Functions:

- Created by the programmer for specific tasks within the program.
- Need to be written before being used in the program.
- **Examples :** `calculateArea(length, width)` - calculates area based on provided values.

Built-in Library Functions :

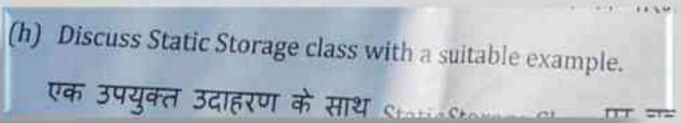
- Can be directly called without defining them in the program.
- Accessible through header files like <stdio.h>, <math.h>, <string.h>, etc.
- **Examples :** `printf()`, `sqrt()`, `strcpy()` - for printing, calculating square root, copying strings, respectively.

Short 6



(g) Define Ternary operator with the help of a suitable example. (1.5)

Short 7

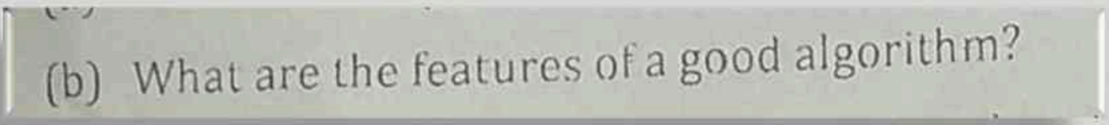

(h) Discuss Static Storage class with a suitable example.

एक उपयुक्त उदाहरण के साथ Static Storage Cl

Static storage in C refers to variables declared with the static keyword. These variables retain their values between function calls, unlike regular local variables that are destroyed after exiting their scope.

2023 Short PYQs

Short 1

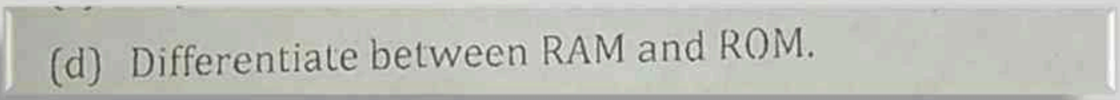

(b) What are the features of a good algorithm?

Correctness: It must produce accurate results for all valid inputs.

Efficiency: It should execute quickly and utilize minimal resources (memory, time).

Clarity: The steps should be clear, concise, and easy to understand for both programmers and others.

Short 2 ➔



(d) Differentiate between RAM and ROM.

RAM (Random Access Memory):

- **Volatile:** Loses data when power is off.
- **Faster:** Provides high-speed access for active programs and data.
- **Temporary:** Stores currently running programs and data.
- **Example:** Stores the code you're writing in your C editor.

ROM (Read-Only Memory):

- **Non-volatile:** Retains data even without power.

- **Slower:** Access speed is slower than RAM.
- **Permanent:** Stores essential instructions for booting and hardware.
- **Example:** BIOS (Basic Input/Output System) on your computer.

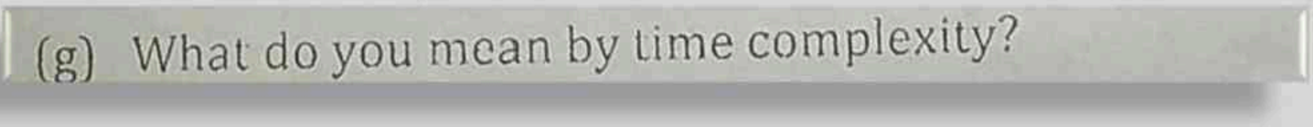
Short 3

(f) What is a header file?

A header file in C is a text file with the extension .h. It acts like a reusable library containing:

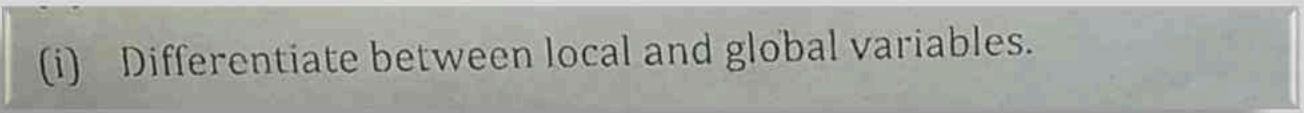
- **Function declarations:** Prototypes outlining function names, parameters, and return types.
- **Macro definitions:** Symbolic constants or shorthand for code snippets.
- **Data type definitions:** Custom data types for specific purposes.

Short 4


(g) What do you mean by time complexity?

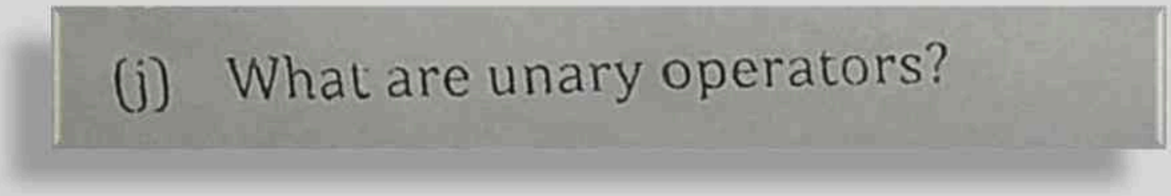
Time complexity in C language refers to the measure of how the runtime of an algorithm grows as the size of the input increases. It quantifies the amount of time an algorithm takes to complete as a function of the input size.

Short 5 ➔


(i) Differentiate between local and global variables.

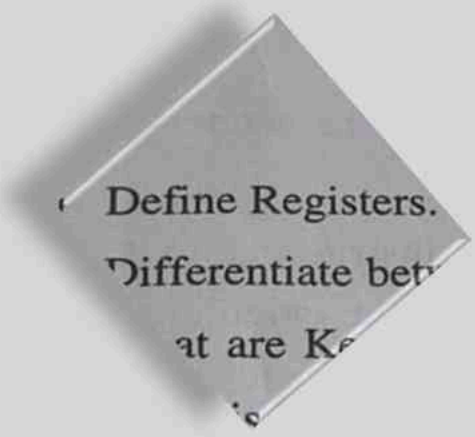
Local variables in C are declared within a block or function and are only accessible within that block or function. They are created when the block or function is entered and destroyed when it exits. Global variables, on the other hand, are declared outside of any function and can be accessed and modified by any function in the program. They have a global scope, meaning they are visible throughout the entire program. Global variables persist throughout the program's execution, while local variables have limited scope and lifetime.

Short 6


(j) What are unary operators?

Unary operators in C are special symbols that operate on a single operand (value) to produce a new value. **Common unary operators include** : Increment (++), Decrement (--) operators, etc.

Short 7

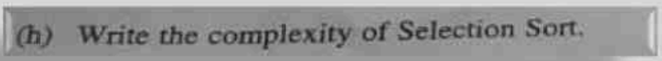

Define Registers.

ifferentiate bet

at are K

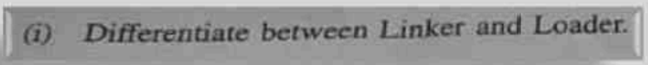
- Registers in C are high-speed memory locations within the CPU. They offer much faster access than regular memory.
- The register keyword suggests to the compiler that a variable should be stored in a register if possible. This can significantly improve performance for frequently used variables.

Short 8


(h) Write the complexity of Selection Sort.

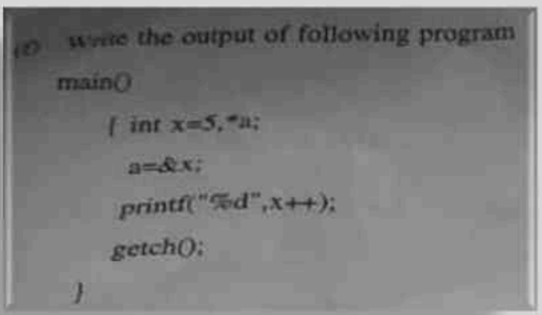
O(n^2)

Short 9


(i) Differentiate between Linker and Loader.

Feature	Linker	Loader
Function	Combines multiple object files into an executable or library	Loads the executable into memory for execution
Activities	Symbol resolution, relocation, and final executable generation	Address binding, relocation, setting up execution environment
Dependency	Operates after the compilation process	Operates after the linking process
Output	Generates a single executable or library file	Loads the executable file into memory
Examples	GNU ld, Microsoft Link, LLVM lld	Windows Loader, Linux ELF loader

Short 10


(j) Write the output of following program

```
main()
    { int x=5,*a;
      a=&x;
      printf("%d",x++);
      getch();
  }
```

5

Explanation:

- **x** is initialized to 5.
- The address of **x** is assigned to pointer **a**.
- **printf("%d",x++);** prints the current value of **x**, which is 5, and then increments **x** by 1. However, the increment operation **x++** doesn't affect the value printed by **printf**. So, 5 is printed.

Roll No. [redacted]

Total Pages : 04

013203

May 2024

B. Tech. (ME/ME(HINDI)/CE/IT/DS/CE(HINDI/ CSE(AIML)) (Second Semester)

Programming for Problem Solving (ESC-103)

Time : 3 Hours] [*Maximum Marks* : 75

Note : It is compulsory to answer all the questions (1.5 marks each) of Part A in short. Answer any *four* questions from Part B in detail. Different sub-parts of a question are to be attempted adjacent to each other.

Part A

1. (a) Define an Algorithm. What are the desirable features of an Algorithm ? **1.5**

(b) Differentiate between Primary Memory and Secondary Memory. **1.5**

(c) What are different types of errors occurred during programming ? **1.5**

(d) What is the difference between Application Software and System Software ? **1.5**

(e) What do you understand by Nesting of If-else ? 1.5

(f) What is Memory bleeding ? 1.5

(g) What is the difference between Library functions and User defined functions ? 1.5

(h) Define a structure. Initialize the member elements of a structure name student with the following values : 1.5

Name-Ramesh, Age-24, Department-CE and Fee is Rs. 34,500.50.

(i) What do you understand by function prototype ? What is its significance ? 1.5

(j) Differentiate between the following using small programming examples :

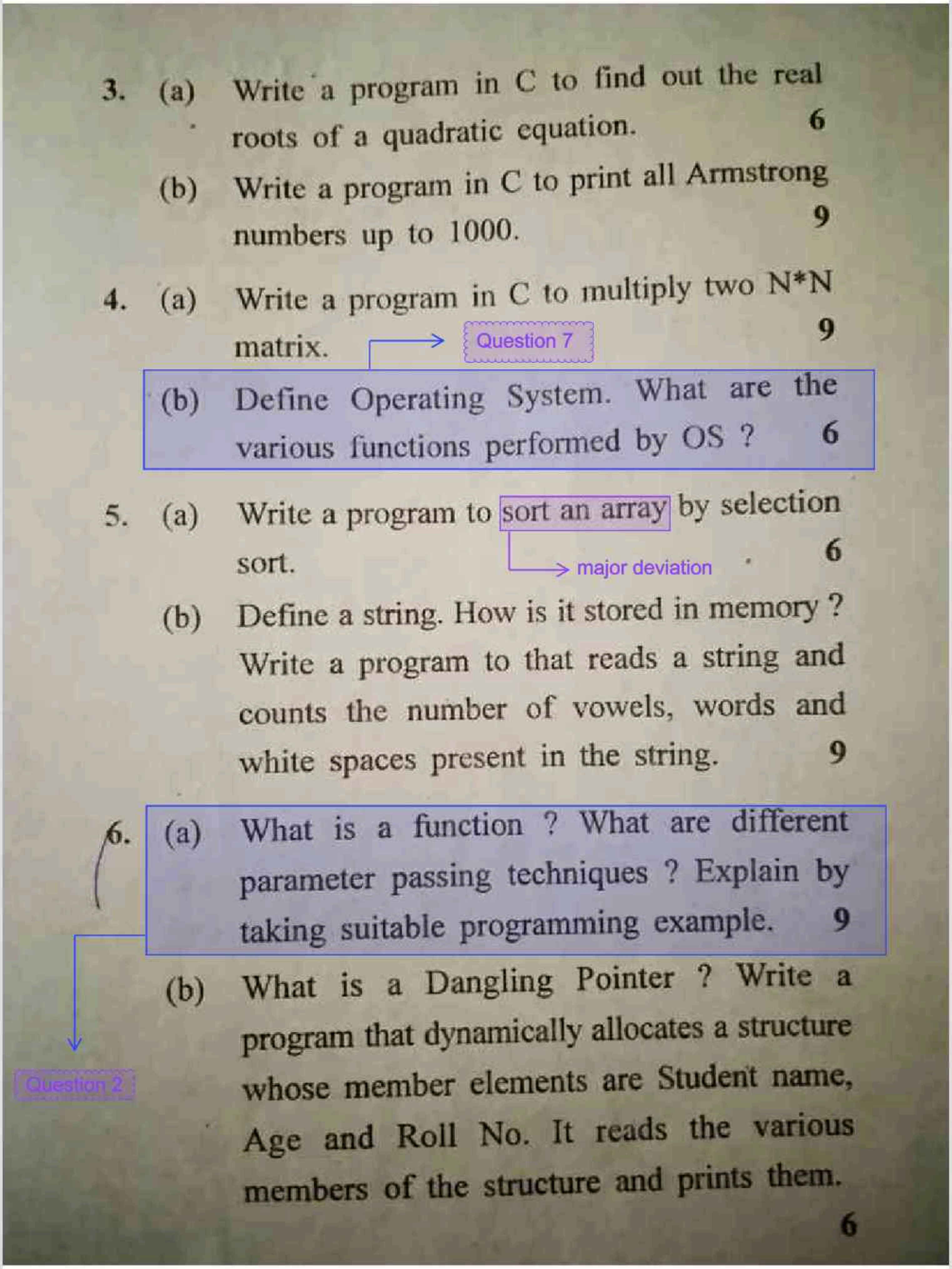
getchar(), gets () and puts(). 1.5

Part B

2. (a) Define Computer System. Explain in detail the functional parts of a computer system with the help of a block diagram. 9

(b) Draw a flow chart to find out the smallest number out of 3 given numbers. 6

Question 1



3. (a) Write a program in C to find out the real roots of a quadratic equation. 6

   (b) Write a program in C to print all Armstrong numbers up to 1000. 9

4. (a) Write a program in C to multiply two N*N matrix. 9

   Question 7

   (b) Define Operating System. What are the various functions performed by OS ? 6

5. (a) Write a program to sort an array by selection sort. 6

   major deviation

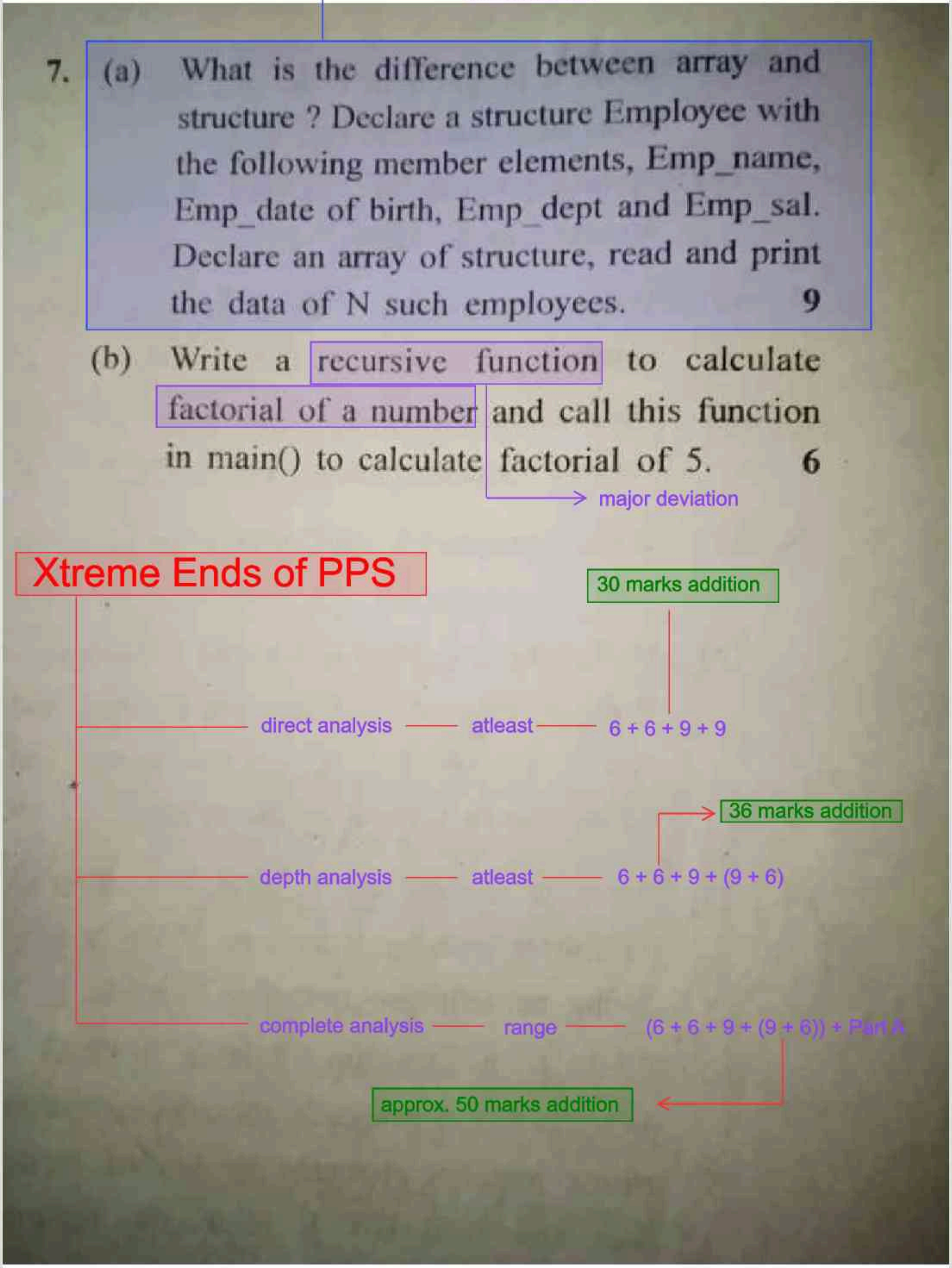
   (b) Define a string. How is it stored in memory ? Write a program to that reads a string and counts the number of vowels, words and white spaces present in the string. 9

6. (a) What is a function ? What are different parameter passing techniques ? Explain by taking suitable programming example. 9

   Question 2

   (b) What is a Dangling Pointer ? Write a program that dynamically allocates a structure whose member elements are Student name, Age and Roll No. It reads the various members of the structure and prints them. 6

Question 8



7. (a) What is the difference between array and structure ? Declare a structure Employee with the following member elements, Emp_name, Emp_date of birth, Emp_dept and Emp_sal. Declare an array of structure, read and print the data of N such employees. 9

(b) Write a recursive function to calculate factorial of a number and call this function in main() to calculate factorial of 5. 6

major deviation

Xtreme Ends of PPS

30 marks addition

direct analysis — atleast — 6 + 6 + 9 + 9

36 marks addition

depth analysis — atleast — 6 + 6 + 9 + (9 + 6)

complete analysis — range — (6 + 6 + 9 + (9 + 6)) + Part A

approx. 50 marks addition

Roll No. 24001007034 Total Pages : 03

007103

December 2024

B.Tech. (First Semester)

Programming for Problem Solving (ESC103)

Time : 3 Hours] [*Maximum Marks* : 75

Note : It is compulsory to answer all the questions (1.5 marks each) of Part A in short. Answer any *four* questions from Part B in detail. Different sub-parts of a question are to be attempted adjacent to each other.

Part A

1. (a) Draw a flowchart to explain the procedure of compilation. 1.5

   (b) What is virtual memory ? Why is it needed ? 1.5

   (c) What are Strings ? Explain the difference between character arrays and strings. 1.5

   (d) What are header files ? 1.5

   (e) What is the difference between cold boot and warm boot ? 1.5

(f) Elaborate a few applications of computers. 1.5

(g) Explain the concept of pointers with the help of an example. 1.5

(h) What do you mean by recursion ? Explain. 1.5

(i) Explain the difference between unary and binary operators. 1.5

(j) How many bytes are required in the memory to store an array int i[20] ? 1.5

Part B

Question 1

2. (a) What is a flowchart ? Draw a flowchart to find the smallest of three numbers. Also write the Algorithm for the same. 10

(b) Draw the block diagram of a computer. Also explain the difference between primary memory and secondary memory. 5

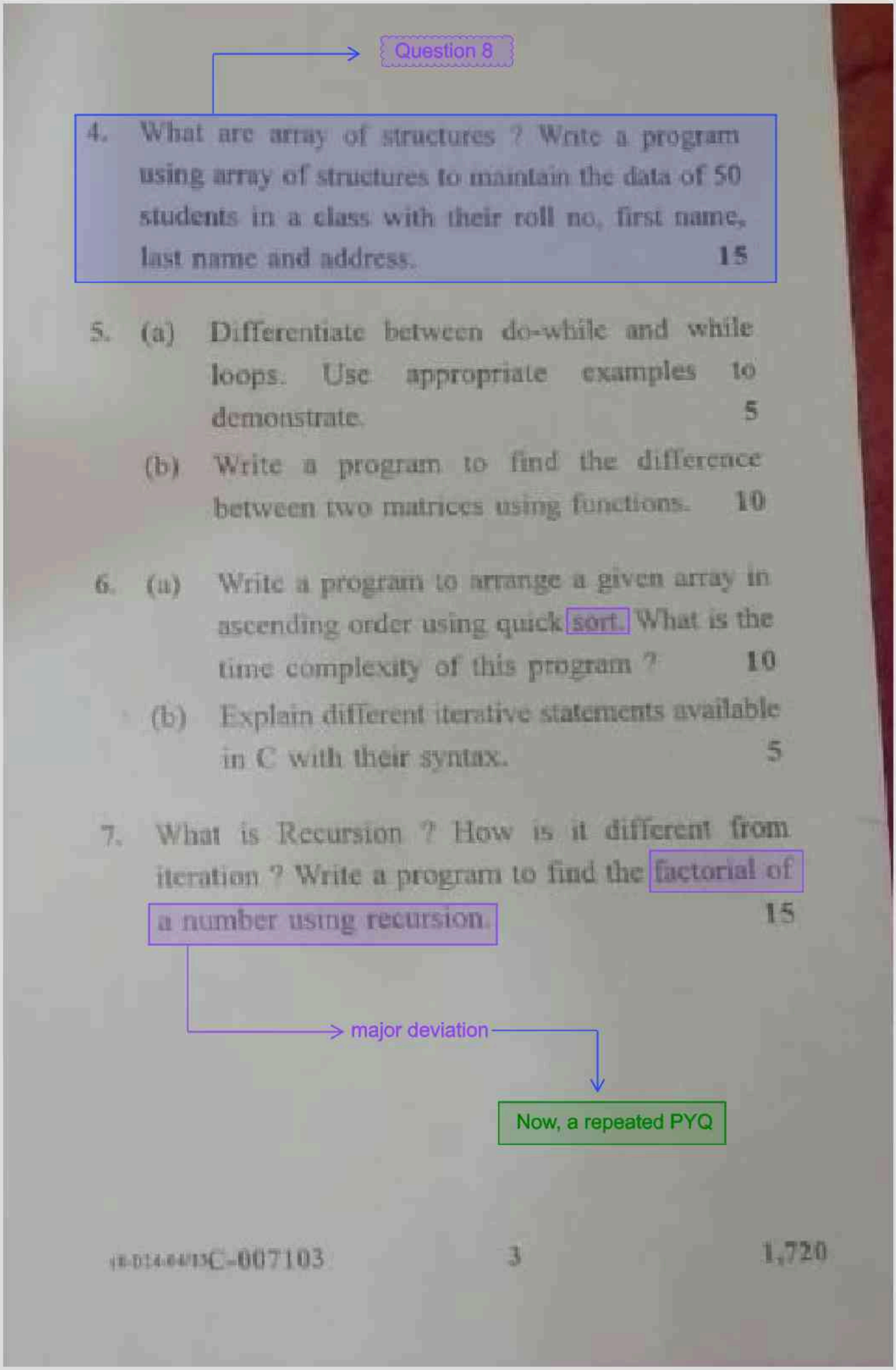
3. (a) Write a program using switch-case to find the grade of a student. 5

(b) Write short notes on the following : 10

(i) Break and continue

(ii) Static variables

(iii) Struct and union.



Question 8

4. What are array of structures ? Write a program using array of structures to maintain the data of 50 students in a class with their roll no, first name, last name and address. 15

5. (a) Differentiate between do-while and while loops. Use appropriate examples to demonstrate. 5

   (b) Write a program to find the difference between two matrices using functions. 10

6. (a) Write a program to arrange a given array in ascending order using quick sort. What is the time complexity of this program ? 10

   (b) Explain different iterative statements available in C with their syntax. 5

7. What is Recursion ? How is it different from iteration ? Write a program to find the factorial of a number using recursion. 15

major deviation

Now, a repeated PYQ