USN | | | | | | | | | | |

Internal Assessment Test 1 – Nov 2025

Sub	Programming in C					Sub code	1BEIT105	Branch	ISE, AIDS, AIML, CSML
Date	06/11/25	Duration	**90 mins**	Max Marks	**50**	Sem /Sec	**I Sem P Cycle (A, B, C, D, E, F, G, H)**		**OBE**

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a) Explain different sections of a C program with suitable example	[5]	CO1	L1
	b) A farmer wants to find the area of his rectangular field. Write a C program to calculate it	[5]	CO1	L3
2	a) Riya, a first-year student, wants to understand how humans communicate with computers. Explain computer languages to her and describe the following with examples — low-level language, high-level language, compiler, and interpreter.	[5]	CO1	L2
	b) Describe how a C program is **created, compiled, and executed** with a diagram.	[5]	CO1	L2
3	a) Predict the output of the following code: `#include <stdio.h>` `int main() { int x = 5, y = 10, z;` `z = x++ + --y * 2;` `printf("%d %d %d", x, y, z);` `return 0;}`	[2]	CO1	L3
	b) Explain about the storage class specifiers with example : 1.auto 2. Register 3. Static 4. extern	[8]	CO1	L2

USN | | | | | | | | | | |

Internal Assessment Test 1 – Nov 2025

Sub	Programming in C					Sub code	1BEIT105	Branch	ISE, AIDS, AIML, CSML
Date	06/11/25	Duration	**90 mins**	Max Marks	**50**	Sem /Sec	**I Sem P Cycle (A, B, C, D, E, F, G, H)**		**OBE**

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a) Explain different sections of a C program with suitable example	[5]	CO1	L1
	b) A farmer wants to find the area of his rectangular field. Write a C program to calculate it	[5]	CO1	L3
2	a) Riya, a first-year student, wants to understand how humans communicate with computers. Explain computer languages to her and describe the following with examples — low-level language, high-level language, compiler, and interpreter.	[5]	CO1	L2
	b) Describe how a C program is **created, compiled, and executed** with a diagram.	[5]	CO1	L2
3	a) Predict the output of the following code: `#include <stdio.h>` `int main() { int x = 5, y = 10, z;` `z = x++ + --y * 2;` `printf("%d %d %d", x, y, z);` `return 0;}`	[2]	CO1	L3
	b) Explain about the storage class specifiers with example : 1.auto 2. Register 3. Static 4. extern	[8]	CO1	L2

4	a) Explain the memory layout of a C program with its segments.	[5]	CO1	L2

	b) A teacher wants to assign grades based on student marks. Explain how selection statements in C can be used for such decisions, with syntax and examples	[5]	CO2	L2
5	Explain the **phases of the System Development Life Cycle (SDLC)** with a neat diagram and detailed explanation of each phase.	[10]	CO1	L2
6	*In a sports event, three players scored different points.* Write a C program to determine who scored the highest using: a) Ternary operator b) if–else if–else statement	[10]	CO2	L3
7	a) Discuss the use of jump statements (break, continue, goto, return) with examples.	[8]	CO2	L2
	b) Predict the output for (int i = 1; i <= 3; i++) { for (int j = 1; j <= 2; j++) { printf("%d%d ", i, j); }}	[2]	CO2	L3
8	a) A shopkeeper wants to store the sales of his shop for a week. Explain how arrays can help in this situation. Write short notes on arrays with syntax and examples for declaration and initialization.	[3]	CO2	L2
	b) Write a C program to find an average of 3 subject marks using the Array concept.	[7]	CO2	L3

CI CCI HOD

4	a) Explain the memory layout of a C program with its segments.	[5]	CO1	L2
	b) A teacher wants to assign grades based on student marks. Explain how selection statements in C can be used for such decisions, with syntax and examples	[5]	CO2	L2
5	Explain the **phases of the System Development Life Cycle (SDLC)** with a neat diagram and detailed explanation of each phase.	[10]	CO1	L2
6	*In a sports event, three players scored different points.* Write a C program to determine who scored the highest using: a) Ternary operator b) if–else if–else statement	[10]	CO2	L3
7	a) Discuss the use of jump statements (break, continue, goto, return) with examples.	[8]	CO2	L2
	b) Predict the output for (int i = 1; i <= 3; i++) { for (int j = 1; j <= 2; j++) { printf("%d%d ", i, j); }}	[2]	CO2	L3
8	a) A shopkeeper wants to store the sales of his shop for a week. Explain how arrays can help in this situation. Write short notes on arrays with syntax and examples for declaration and initialization.	[3]	CO2	L2
	b) Write a C program to find an average of 3 subject marks using the Array concept.	[7]	CO2	L3

CI CCI HOD

Solution with scheme-Model Answer

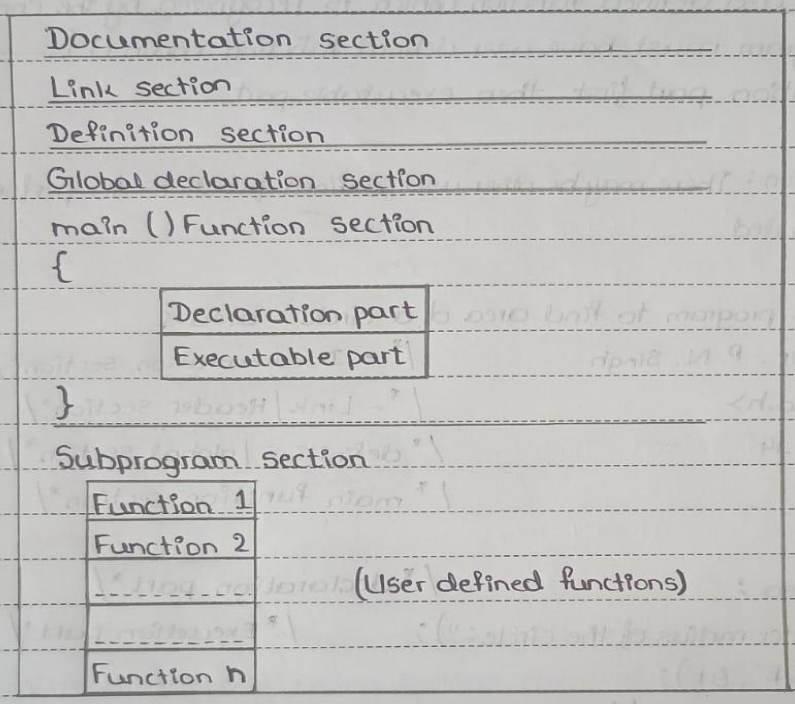
Prof. Rajni Tiwari, Prof.Reshma, Prof. Dhivya, Prof. Saumya, Prof. kiran, Dr Sivasakti, Prof Shilpashree									
Internal Assessment Test 1 – November 2025									
Sub	**Programming in C**					Sub code	**1BEIT105**	Branch	**ISE, AIDS, AIML, CSML**
Date	**6.11.2025**	Duration	**90 mins**	Max Marks	**50**	Sem /Sec	**I Sem P Cycle (A, B, C, D, E, F, G, H)**		**OBE**

	Answer any FIVE FULL Questions	**MARKS**	**CO**	**RBT**
1	**a) Explain different sections of a C program with suitable example**	**[5]**	**CO1**	**L1**
	b) A farmer wants to find the area of his rectangular field. Write a C program to calculate it	**[5]**	**CO1**	**L3**
2	**a) Riya, a first-year student, wants to understand how humans communicate with computers. Explain computer languages to her and describe the following with examples — low-level language, high-level language, compiler, and interpreter.**	**[5]**	**CO1**	**L2**
	b) Describe how a C program is created, compiled, and executed with a diagram.	**[5]**	**CO1**	**L2**
3	**a) Predict the output of the following code:** `#include <stdio.h>` `int main() { int x = 5, y = 10, z;` `z = x++ + --y * 2;` `printf("%d %d %d", x, y, z);` `return 0;}`	**[2]**	**CO1**	**L3**
	b) Explain about the storage class specifiers with example : 1.auto 2. Register 3. Static 4. extern	**[8]**	**CO1**	**L2**
4	**a) Explain the memory layout of a C program with its segments.**	**[5]**	**CO1**	**L2**
	b) A teacher wants to assign grades based on student marks. Explain how selection statements in C can be used for such decisions, with syntax and examples	**[5]**	**CO2**	**L2**
5	**Explain the phases of the System Development Life Cycle (SDLC) with a neat diagram and detailed explanation of each phase.**	**[10]**	**CO1**	**L2**
6	***In a sports event, three players scored different points.*** **Write a C program to determine who scored the highest using:** **a) Ternary operator b) if–else if–else statement**	**[10]**	**CO2**	**L3**
7	**a) Discuss the use of jump statements (break, continue, goto, return) with examples.**	**[8]**	**CO2**	**L2**
	b) Predict the output `for (int i = 1; i <= 3; i++) {` `for (int j = 1; j <= 2; j++) {` `printf("%d%d ", i, j);` `}}`	**[2]**	**CO2**	**L3**
8	**a) A shopkeeper wants to store the sales of his shop for a week. Explain how arrays can help in this situation. Write short notes on arrays with syntax and examples for declaration and initialization.**	**[3]**	**CO2**	**L2**
	b) Write a C program to find an average of 3 subject marks using the Array concept.	**[7]**	**CO2**	**L3**

1. **a) Explain different sections of a C program with suitable example** [5]

min 4 section [4] +example [1]

1) Explain the basic structure of a C program with example.



• Documentation Section: This section is used to write problem, file n developer, date etc in comment lines within /*....*/ or separate line com may start with //. Compiler ignores this section. Documentation enhance the readability of a program.

• Link section: To include header and library files whose in-built struct are to be used. Linker also required these files to build a program executable. Files are included with directive, #include

- Definition section : To define macros and symbolic constants by preprocessor directive #define.

- Global section: to declare global variables - to be accessed by all functions.

- main() is the user defined function which is recognized by the compiler first. So, all C program must have user defined function main () {.....}. It should have declaration part first then executable part.

- Sub program section : There may be other user defined functions to perform specific task when called

```
/*Example : a program to find area of a circle - area.c
            Dr. P. N. Singh                    /* - Documentation section*/
#include <stdio.h>                             /* - Link/Header section*/
#define PI 3.14                                /* definition/global section*/
int main()                                     /* main function section*/
{
    float r, area;                             /* declaration part*/
    printf("Enter radius of the circle:");     /* Execution part*/
    scanf("%f", &r);
    area = PI*r*r;  /*using symbolic constant PI*/
    printf("Area of circle = %0.3f square unit \n", area);
    return (0);
}
```

b) A farmer wants to find the area of his rectangular field. Write a C program to calculate it

Full Program [5]

Solution:

```
#include <stdio.h>

int main() {
    float length, width, area;

    // Input the length and width of the field
    printf("Enter the length of the rectangular field (in meters): ");
    scanf("%f", &length);

    printf("Enter the width of the rectangular field (in meters): ");
    scanf("%f", &width);

    // Calculate the area
    area = length * width;

    // Display the result
    printf("The area of the rectangular field is: %.2f square meters\n", area);

    return 0;
}
```

[5]

1

[5]

2.

a) Riya, a first-year student, wants to understand how humans communicate with computers. Explain computer languages to her and describe the following with examples — low-level language, high-level language, compiler, and interpreter.

Explanation [2] + Types [3]

[5]

Solution:

Humans and computers don't speak the same language.

- Humans use natural languages like **English or Hindi**,
- But computers only understand **machine language (0s and 1s)**.

To make communication possible, we use **computer languages** — special languages that humans can write instructions in, which computers can then understand and execute.

Types of Computer Languages

1. Low-Level Language

- These are languages that are **close to machine language**.
- They are **difficult for humans** to read and understand but are **fast for computers** to execute.
- Two main types:
  - **Machine Language:** Written using only 0s and 1s (binary code).
    Example:

10110000 01100001

Assembly Language: Uses short words (mnemonics) instead of binary.
Example: MOV A, B

2. High-Level Language

- These are **closer to human languages** like English.
- Easy to read, write, and understand.
- A **translator (compiler or interpreter)** converts them into machine code.

Examples: C, C++, Python, Java
Example in **C language**:

```
printf("Hello, World!");
```

Translators: How High-Level Code Becomes Machine Code

1. Compiler

- A **compiler** translates the **entire program** at once into machine code.
- If there are errors, it shows them all **after compilation**.
- Example languages: **C, C++**

2. Interpreter

- An **interpreter** translates and executes the program **line by line**.
- It stops translating when it finds an error.

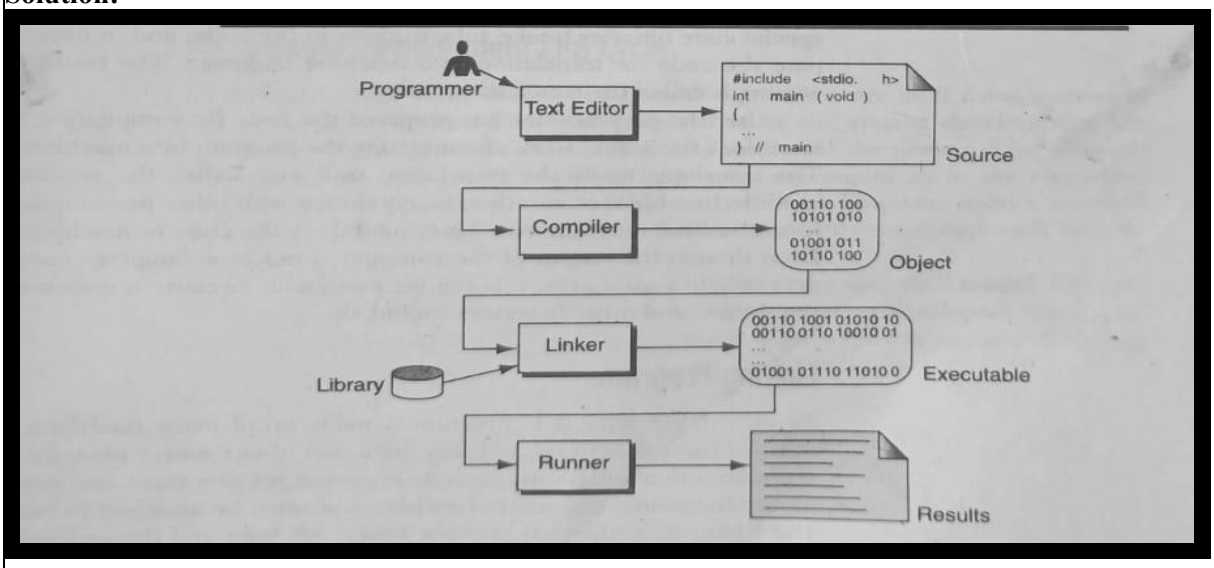
Example languages: Python, JavaScript

[5]

b) Describe how a C program is created, compiled, and executed with a diagram.

Diagram [2] + explanation [3]

Solution:



[5]

Steps involved in this process are:

1. **Editing: Write code using a text editor and save with .c extension.**
2. **Compilation: Compiler translates source code into object code.**
3. **Linking: Linker combines object code with library functions.**

Loading & Execution: Loader loads the executable into memory for execution.

[10]

3.

a) Predict the output of the following code:

```
#include <stdio.h>
int main() { int x = 5, y = 10, z;
   z = x++ + --y * 2;
   printf("%d %d %d", x, y, z);
   return 0;}
```

[2]

Output [2]

Solution:

6 9 23

b)Explain about the storage class specifiers with example : 1.auto 2. Register 3. Static 4. Extern [8]

each point [2]

Solution:
Storage classes define the scope, lifetime, and visibility of variables.

1. **Keyword: auto**

Scope: Local to the function or block where it is declared.
Lifetime: Exists till the function execution completes.
Default Type: If no storage class is mentioned for a local variable, it is automatically auto. And will have garbage value
Storage Location: RAM stack segment
Example:

```
void display() {
   auto int a = 10;
   printf("%d", a);
}
```

2. Keyword: register
Scope: Local to the function or block where it is declared.
Lifetime: Exists until the function is executed; memory is freed once the function ends.
Default Type: Garbage
Storage Location:CPU registers

Example:

```
void show() {
   register int i;
   for(i=0; i<5; i++)
      printf("%d ", i);
}
```

3. Keyword: static
Scope: Local to the block, but value is retained between function calls/ Global
Lifetime: Throughout the program execution.
Default Type: 0 for int, depends on data type
Storage Location: RAM static/Global segment
Example:

```
void counter() {
   static int count = 0;
   count++;
   printf("%d ", count);
}
```

4. Keyword: extern (basically used in two files)
Scope: Global, accessible by all functions in the program.
Lifetime: Entire program execution.
Default Type: 0 for int, depends on data type
Storage Location: RAM global segment
Example:

```
int num = 10; // global variable

void printNum() {
   extern int num;  // refers to the global variable
   printf("%d", num);
}
```

4 **a) Explain the memory layout of a C program with its segments.** [5]

Diagram [1] + explanation of 4 segment [4]

Solution:

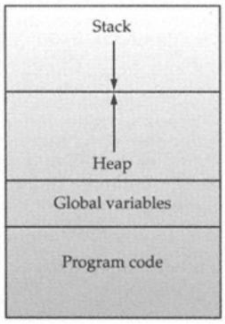


Figure 1-2
Conceptualized memory
map of a C program

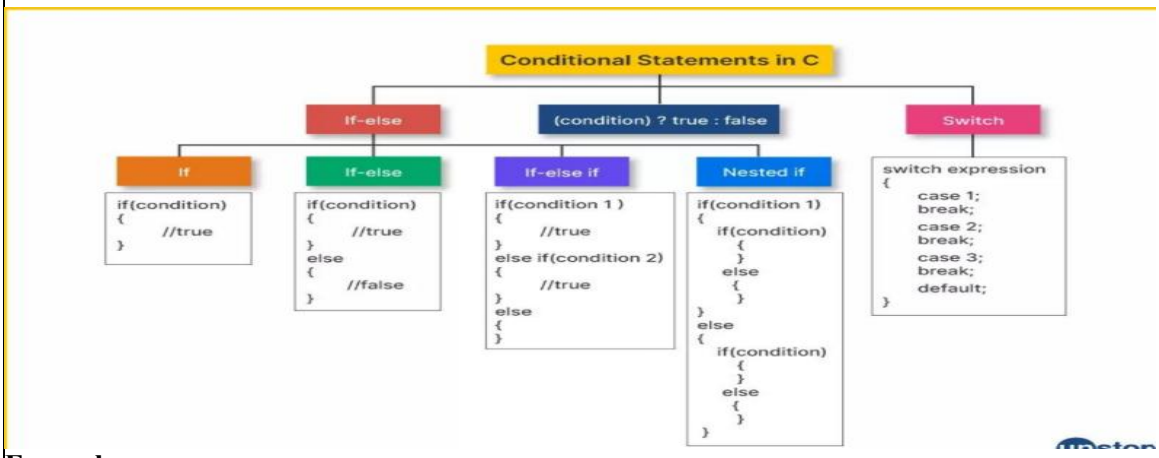
1. **Text Segment: Stores executable code.**
2. **Data Segment: Stores initialized global/static variables.**
3. **BSS Segment: Stores uninitialized global/static variables.**
4. **Heap: Dynamic memory allocated using malloc()/calloc().**
5. **Stack: Stores local variables and function call info.**

b) A teacher wants to assign grades based on student marks. Explain how selection statements in C can be used for such decisions, with syntax and examples

syntax and explanation of all 5 type [5]

Solution:

[5]



Example:

1. if Statement

```
if (a > b)
        printf("a is greater");
```

2. if–else Statement

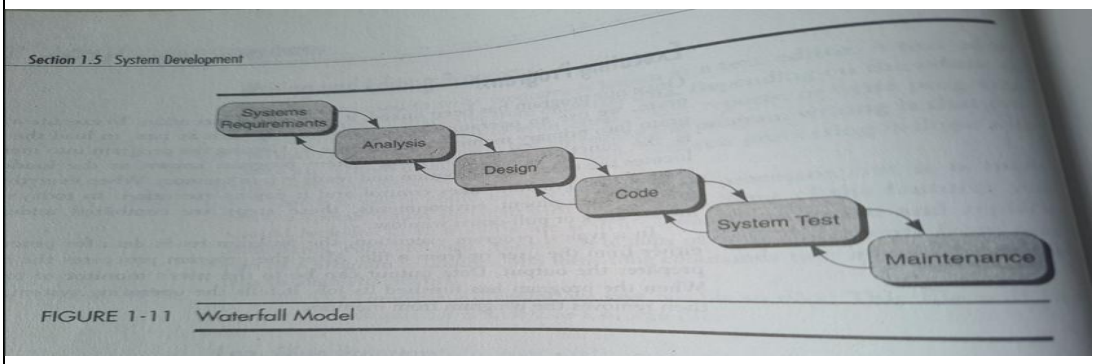
```
if (x % 2 == 0)
        printf("Even");
else
        printf("Odd");
```

3. Nested if

```
if(a > b)
  if(a > c)
    printf("a is largest");
```

4. switch Statement

```
switch(choice) {
  case 1: printf("Add"); break;
  case 2: printf("Subtract"); break;
  default: printf("Invalid");
}
```

5	**Explain the phases of the System Development Life Cycle (SDLC) with a neat diagram and detailed explanation of each phase.** **Diagram [5] + explanation [5]**  FIGURE 1-11 Waterfall Model **1. Requirement Analysis** **In this phase, developers and analysts work with users to understand and document what the system should do. All functional and non-functional requirements are collected through interviews or discussions and recorded in a Software Requirement Specification (SRS) document.** ***Example:*** **Deciding what features a student management system should have, like adding, viewing, or updating student details.** **2. System Design** **This phase focuses on how the system will work. Designers create a blueprint showing the system architecture, data flow, database structure, and user interface. It helps developers know what to build.** ***Example:*** **Designing tables for students, subjects, and marks, and creating a layout for the input forms.** **3. Implementation (Coding)** **Developers now write the actual source code using suitable programming languages based on the design. Each module is coded and then combined to form the complete system.** ***Example:*** **Writing the program in C, Java, or Python to store and display student data.** **4. Testing** **After coding, the system is tested to find and fix errors. Different testing methods ensure that the software meets requirements and works correctly under all conditions.** ***Example:*** **Checking if the system correctly saves and retrieves student details.** **5. Deployment** **Once testing is successful, the software is installed and delivered to the user environment. Users start using the system, and minor adjustments may be made based on feedback.** ***Example:*** **Installing the student management software in a school computer lab.** **6. Maintenance** **After deployment, the system requires regular updates and bug fixes to adapt to new user needs or technologies. This phase ensures the software remains reliable and efficient over time.** ***Example:*** **Adding new features like attendance tracking or online access later.**	[10]

6 ***In a sports event, three players scored different points.*** **Write a C program to determine who scored the highest using:**

a) Ternary operator b) if–else if–else statement

Program a) [5] + Program b) [5]

Solution:

(a) Using Ternary Operator [10]

```
#include <stdio.h>

int main() {
   int p1, p2, p3, highest;

   // Input scores
   printf("Enter the score of Player 1: ");
   scanf("%d", &p1);
   printf("Enter the score of Player 2: ");
   scanf("%d", &p2);
   printf("Enter the score of Player 3: ");
   scanf("%d", &p3);

   // Using nested ternary operator to find highest
   highest = (p1 > p2) ? ((p1 > p3) ? p1 : p3) : ((p2 > p3) ? p2 : p3);

   // Display result
   printf("The highest score is: %d\n", highest);

   return 0;
}
```

(b) Using if–else if–else Statement

```
#include <stdio.h>

int main() {
   int p1, p2, p3;

   // Input scores
   printf("Enter the score of Player 1: ");
   scanf("%d", &p1);
   printf("Enter the score of Player 2: ");
   scanf("%d", &p2);
   printf("Enter the score of Player 3: ");
   scanf("%d", &p3);

   // Using if–else if–else to find the highest
   if (p1 > p2 && p1 > p3) {
      printf("Player 1 scored the highest with %d points.\n", p1);
   }
   else if (p2 > p1 && p2 > p3) {
      printf("Player 2 scored the highest with %d points.\n", p2);
   }
   else {
      printf("Player 3 scored the highest with %d points.\n", p3);
   }

   return 0;
}
```

7	**a) Discuss the use of jump statements (break, continue, goto, return) with examples.** **Each statement with example [2]** **Solution:** **Jump statements alter normal flow of control.** **1. break: Exit from loop/switch.** **Example:** **for(i=0;i<5;i++){** **if(i==3)** **break;** **printf("%d ",i);** **}** **2. continue: Skip rest of loop iteration.** **Example:** **for(i=0;i<5;i++){** **if(i==3)** **continue;** **printf("%d ",i);** **}** **3. goto: Jump to labeled statement.** **Example:** **Void main()** **{** **printf("I am student1");** **goto L;** **printf("I am student2");** **L: printf("I am student3");** **}** **4. return: Exit from function.** Syntax: return expression; Example: return 0; // successful termination of program return 1 & return -1 ; // Unsuccessful termination of program	[8]

	b) Predict the output **for (int i = 1; i <= 3; i++) {** **for (int j = 1; j <= 2; j++) {** **printf("%d%d ", i, j);** **}}** **Proper output [2]** **Solution:** **11 12 21 22 31 32**	[2]

8	**a) A shopkeeper wants to store the sales of his shop for a week. Explain how arrays can help in this situation. Write short notes on arrays with syntax and examples for declaration and initialization.** **Array definition [1] +declaration [1] + initialization [1]** **Solution:** **An array is a collection of similar data types stored in contiguous memory locations.** **It allows storing multiple values under a single variable name, making it easier to manage large sets of data.** **Key Points:** • **All elements in an array must be of the same data type (e.g., all integers or all floats).** • **Each element is accessed using an index, which starts from 0.** • **Arrays help in storing and processing large data efficiently using loops.** **Declaration Syntax:** **Data_type name_of_Array[size];** **Example:** **int marks[5];** **Initialization:** **int marks[5] = {50, 60, 70, 80, 90};** **Or** **int marks[ ] = {50, 60, 70, 80, 90};** **Or** **marks[0]= 50;** **marks[1]= 60; and so on** **b) Write a C program to find an average of 3 subject marks using the Array concept.** **Full Program [7]** **Solution:** **#include <stdio.h>** **int main() {** **int marks[3]; // Array to store marks of 3 subjects** **int i, sum = 0;** **float average;** **// Input marks** **printf("Enter marks of 3 subjects:\n");** **for(i = 0; i < 3; i++) {** **printf("Subject %d: ", i + 1);** **scanf("%d", &marks[i]);** **sum += marks[i]; // Add marks to sum** **}** **// Calculate average** **average = sum / 3.0;** **// Display result** **printf("\nAverage marks = %.2f\n", average);** **return 0;** **}**	[3] [7]