

# Programming In C (4331105) - Winter 2022 Solution

Milav Dabgar

March 28, 2023

## Question 1 [a marks]

3 List basic data types of C language with their range

### Solution

Answer:

Data Type	Size (bytes)	Range
char	1	-128 to 127
int	2 or 4	-32,768 to 32,767 (2 bytes) or -2,147,483,648 to 2,147,483,647 (4 bytes)
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308

### Mnemonic

"CIFD - Computer Is Fundamentally Digital"

## Question 1 [b marks]

4 Explain rules for naming a variable.

### Solution

Answer:

Rule	Example
Must start with letter or underscore	valid: <code>_count</code> , <code>name</code> / invalid: <code>1score</code>
Can contain letters, digits, underscores	valid: <code>user_1</code> / invalid: <code>user-1</code>
Cannot use keywords	valid: <code>integer</code> / invalid: <code>int</code>
Case sensitive	<code>total</code> and <code>TOTAL</code> are different

Diagram:

### Variable Naming Rules

$[A-Z, a-z, \_] \rightarrow [A-Z, a-z, 0-9, \_]^*$

### Mnemonic

"LUCK - Letters Underscore Case Keywords"

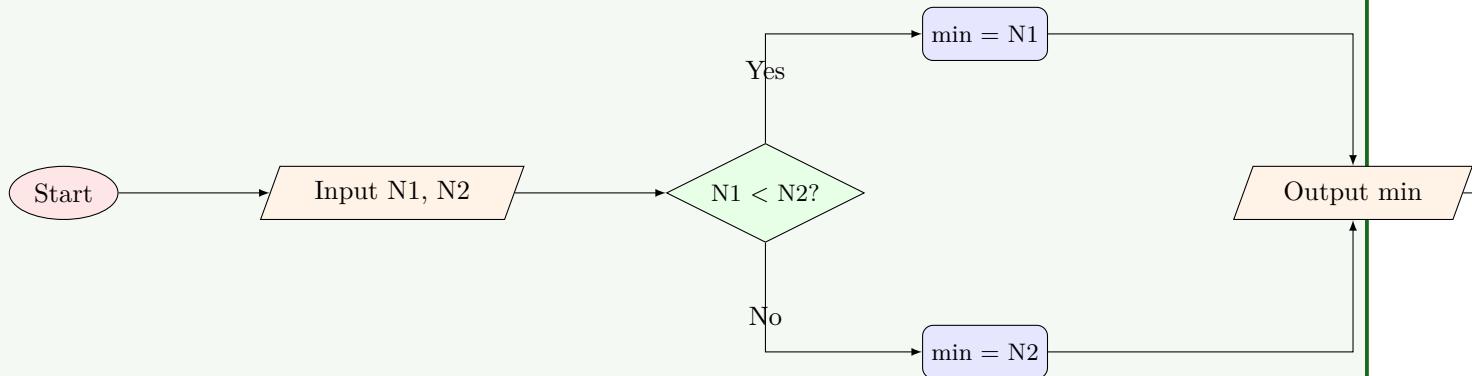
## Question 1 [c marks]

7 Define flowchart. Draw flowchart to find minimum of two integer numbers N1 and N2.

### Solution

#### Answer:

A flowchart is a graphical representation of an algorithm showing the sequence of steps using standard symbols connected by arrows.



- **Flowchart symbols:** Visual representation of logical steps
- **Decision diamond:** Tests condition to determine flow path
- **Process boxes:** Contain calculations or operations

### Mnemonic

"FAST - Flow Analysis Shown Through-charts"

OR

## Question 1 [c marks]

7 Define algorithm. Write an algorithm to calculate area and circumference of circle.

### Solution

#### Answer:

An algorithm is a step-by-step procedure to solve a particular problem in a finite sequence of well-defined instructions.

#### Algorithm for circle calculations:

1. START
2. Input radius r
3. Calculate area = PI \* r \* r
4. Calculate circumference = 2 \* PI \* r
5. Output area, circumference
6. STOP

Step	Operation	Formula
1	Get radius	Input r
2	Calculate area	$A = \pi \times r^2$
3	Calculate circumference	$C = 2 \times \pi \times r$
4	Display results	Output A, C

**Mnemonic**

"SICS - Steps In Clear Sequence"

**Question 2 [a marks]**

3 Differentiate printf() and scanf().

**Solution****Answer:**

Feature	printf()	scanf()
Purpose	Outputs data to screen	Inputs data from keyboard
Format	<code>printf("format", variables)</code>	<code>scanf("format", &amp;variables)</code>
Returns	Number of chars printed	Number of items successfully read
Addressing	Uses variable names	Uses address of variables (&var)

**Mnemonic**

"IO-AR - Input Output-Address Returns"

**Question 2 [b marks]**

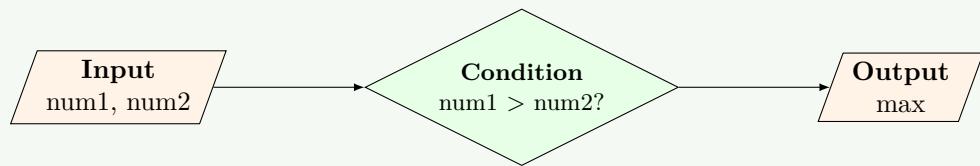
4 Develop a C program to find maximum among two numbers using conditional operator.

**Solution****Answer:**

```

1 #include <stdio.h>
2
3 int main() {
4     int num1, num2, max;
5
6     printf("Enter two numbers: ");
7     scanf("%d %d", &num1, &num2);
8
9     max = (num1 > num2) ? num1 : num2;
10
11    printf("Maximum number is: %d", max);
12
13    return 0;
14 }
```

**Diagram:**

**Mnemonic**

"CTO - Condition Then Output"

**Question 2 [c marks]**

7 Explain arithmetic & relational operators with examples.

**Solution****Answer:**

Type	Operators	Example	Result
<b>Arithmetic Operators</b>			
Addition	+	5 + 3	8
Subtraction	-	5 - 3	2
Multiplication	*	5 * 3	15
Division	/	5 / 3	1 (integer division)
Modulus	%	5 % 3	2 (remainder)
<b>Relational Operators</b>			
Equal to	==	5 == 3	0 (false)
Not equal to	!=	5 != 3	1 (true)
Greater than	>	5 > 3	1 (true)
Less than	<	5 < 3	0 (false)
Greater than or equal	>=	5 >= 5	1 (true)
Less than or equal	<=	5 <= 3	0 (false)

**Mnemonic**

"ASMDCRO - Add Subtract Multiply Divide Compare Return Output"

**OR**

**Question 2 [a marks]**

3 Considering precedence of operators, write down each step of evaluation and final answer if expression  $(25/3) * 4 - 10 \% 3 + 9/2$  is evaluated.

**Solution****Answer:**

Step	Operation	Calculation	Result
1	Parentheses $(25/3)$	$25/3 = 8$ (integer division)	8
2	Modulus $10 \% 3$	$10 \% 3 = 1$	1
3	Division $9/2$	$9/2 = 4$ (integer division)	4
4	Multiplication $8 * 4$	$8 * 4 = 32$	32
5	Subtraction $32 - 1$	$32 - 1 = 31$	31
6	Addition $31 + 4$	$31 + 4 = 35$	35

Final answer = 35

### Mnemonic

"PEMDAS - Parentheses, Exponents, Multiplication/Division, Addition/Subtraction"

OR

## Question 2 [b marks]

4 Develop a C program to find roots of an algebraic equation

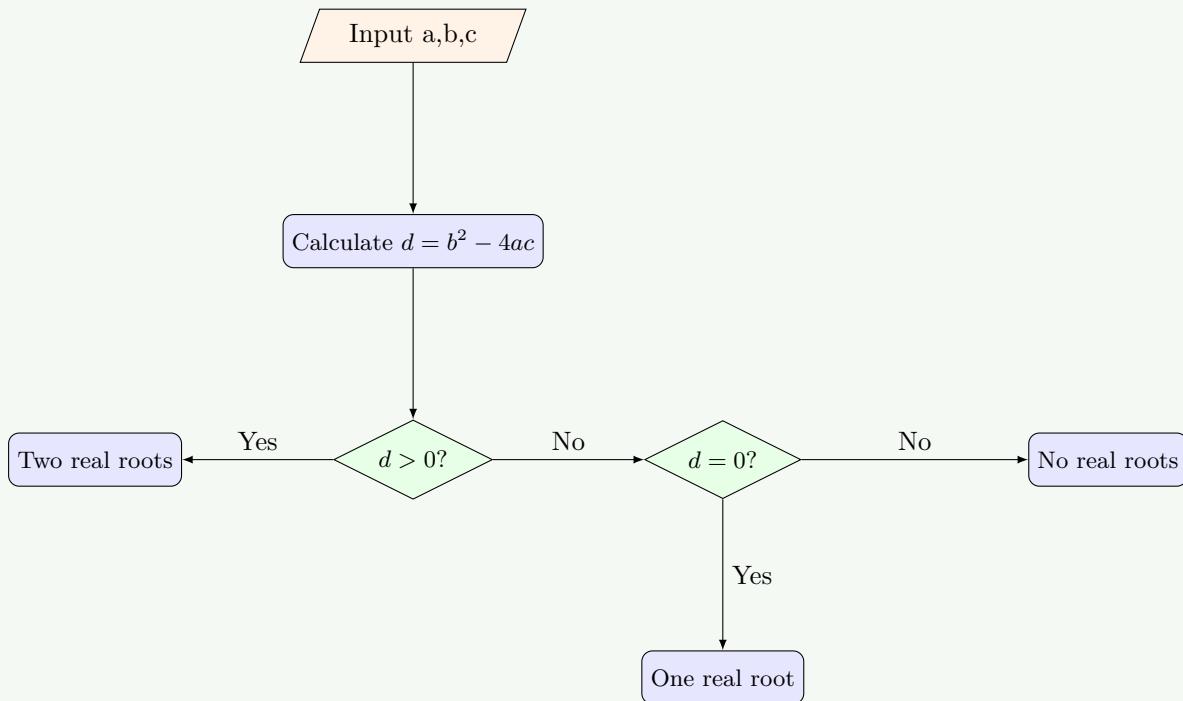
### Solution

Answer:

```

1 #include <stdio.h>
2 #include <math.h>
3
4 int main() {
5     float a, b, c;
6     float discriminant, root1, root2;
7
8     printf("Enter coefficients a, b, c: ");
9     scanf("%f %f %f", &a, &b, &c);
10
11    discriminant = b*b - 4*a*c;
12
13    if (discriminant > 0) {
14        root1 = (-b + sqrt(discriminant)) / (2*a);
15        root2 = (-b - sqrt(discriminant)) / (2*a);
16        printf("Roots: %.2f and %.2f", root1, root2);
17    } else if (discriminant == 0) {
18        root1 = -b / (2*a);
19        printf("Root: %.2f", root1);
20    } else {
21        printf("No real roots");
22    }
23
24    return 0;
25 }
```

Diagram:

**Mnemonic**

"QDR - Quadratic Discriminant Roots"

**OR**

**Question 2 [c marks]**

7 Explain logical & bit-wise operators with examples.

**Solution****Answer:**

Type	Operators	Example	Result
<b>Logical Operators</b>			
Logical AND	&&	(5>3) && (4<7)	1 (true)
Logical OR		(5<3)    (4<7)	1 (true)
Logical NOT	!	!(5>3)	0 (false)
<b>Bitwise Operators</b>			
Bitwise AND	&	5 & 3 (101 & 011)	1 (001)
Bitwise OR		5   3 (101   011)	7 (111)
Bitwise XOR	^	5 ^ 3 (101 ^ 011)	6 (110)
Bitwise NOT	~	~5 (~ 00000101)	-6 (11111010)
Left Shift	«	5 « 1 (101 « 1)	10 (1010)
Right Shift	»	5 » 1 (101 » 1)	2 (10)

**Mnemonic**

"LAND BORNs - Logical AND OR NOT, Bitwise OR AND NOT Shift"

## Question 3 [a marks]

3 Explain the use of 'go to' statement with example

### Solution

#### Answer:

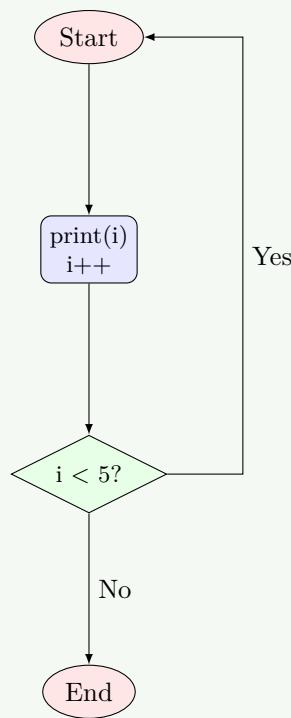
The goto statement allows unconditional jump to a labeled statement in the program.

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 0;
5
6     start:
7         printf("%d ", i);
8         i++;
9         if (i < 5)
10             goto start;
11
12     return 0;
13 }
// Output: 0 1 2 3 4

```

#### Diagram:



#### Mnemonic

"JUMP - Just Unconditionally Move Program-counter"

## Question 3 [b marks]

4 Develop a C program to check whether the entered number is even or odd.

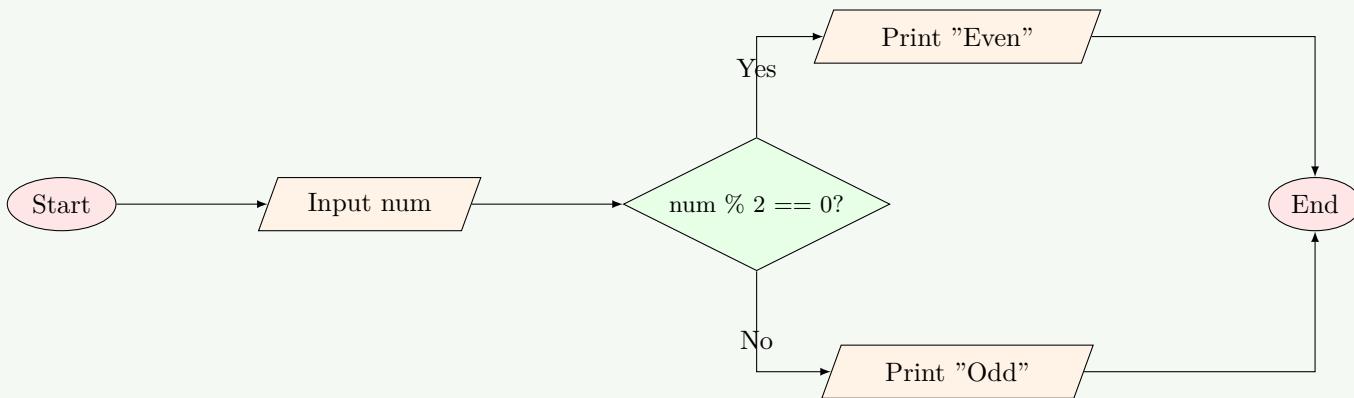
## Solution

### Answer:

```

1 #include <stdio.h>
2
3 int main() {
4     int num;
5
6     printf("Enter a number: ");
7     scanf("%d", &num);
8
9     if (num % 2 == 0)
10        printf("%d is even", num);
11    else
12        printf("%d is odd", num);
13
14    return 0;
15 }
```

### Diagram:



### Mnemonic

"MODE - Modulo Odd-Even Determination"

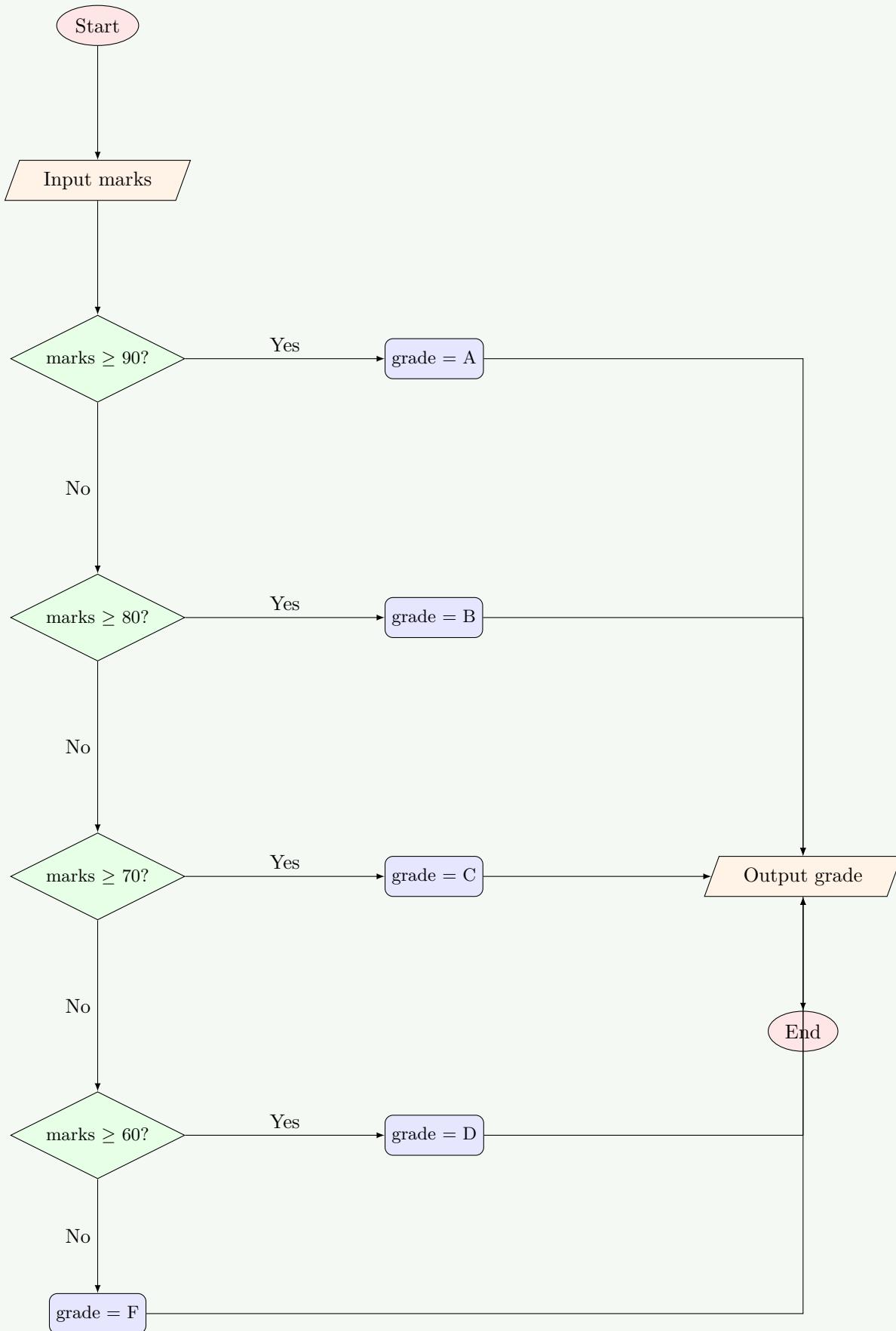
## Question 3 [c marks]

7 Draw flowchart and explain else if ladder with example.

## Solution

### Answer:

The else-if ladder allows checking multiple conditions in sequence, executing the block associated with the first true condition.



```

1 #include <stdio.h>
2
3 int main() {
4     int marks;
5     char grade;
6
7     printf("Enter marks: ");
8     scanf("%d", &marks);
9
10    if (marks >= 90)
11        grade = 'A';
12    else if (marks >= 80)
13        grade = 'B';
14    else if (marks >= 70)
15        grade = 'C';
16    else if (marks >= 60)
17        grade = 'D';
18    else
19        grade = 'F';
20
21    printf("Grade: %c", grade);
22
23    return 0;
24 }
```

- **Multiple conditions:** Checks conditions sequentially
- **First match:** Only executes code for first true condition
- **Default case:** Final else handles all remaining cases

#### Mnemonic

"CAFE - Condition Assess First Eligible"

OR

## Question 3 [a marks]

3 Explain the use of continue and break statement.

#### Solution

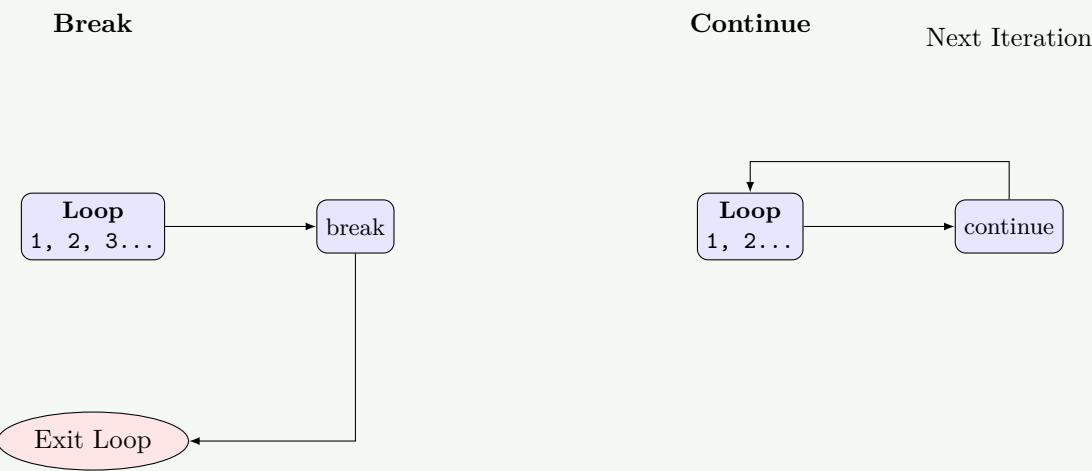
##### Answer:

Statement	Purpose	Effect
break	Exit a loop or switch	Terminates entire loop immediately
continue	Skip current iteration	Jumps to next iteration of loop

```

1 // break example
2 for(int i=1; i<=10; i++) {
3     if(i == 6)
4         break;      // Exits loop when i=6
5     printf("%d ", i); // Output: 1 2 3 4 5
6 }
7
8 // continue example
9 for(int i=1; i<=10; i++) {
10    if(i % 2 == 0)
11        continue;  // Skips even numbers
12    printf("%d ", i); // Output: 1 3 5 7 9
13 }
```

**Diagram:**



### Mnemonic

"BEST - Break Exits, Skip with conTinue"

OR

## Question 3 [b marks]

4 Develop a C program to print sum of 1 to 10 numbers using for loop.

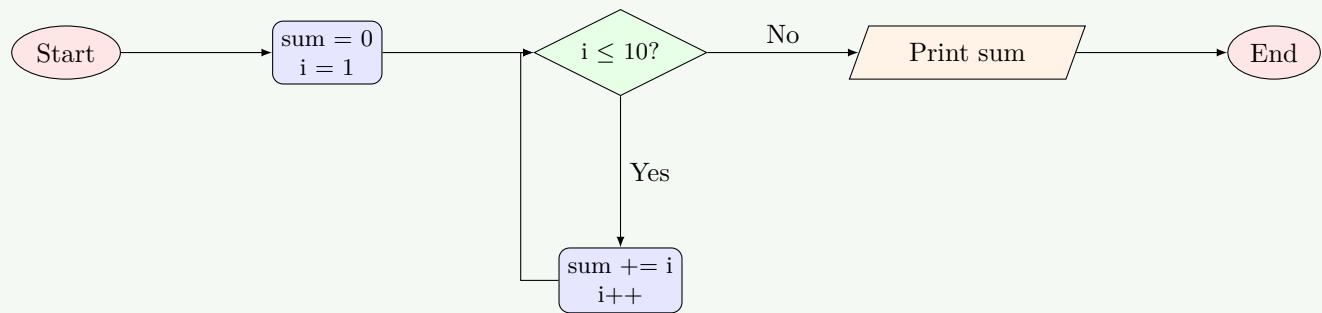
### Solution

#### Answer:

```

1 #include <stdio.h>
2
3 int main() {
4     int i, sum = 0;
5
6     for(i = 1; i <= 10; i++) {
7         sum += i;
8     }
9
10    printf("Sum of numbers from 1 to 10: %d", sum);
11
12    return 0;
13 }
```

**Diagram:**



**Mnemonic**

"SILA - Sum Increment Loop Add"

**OR**

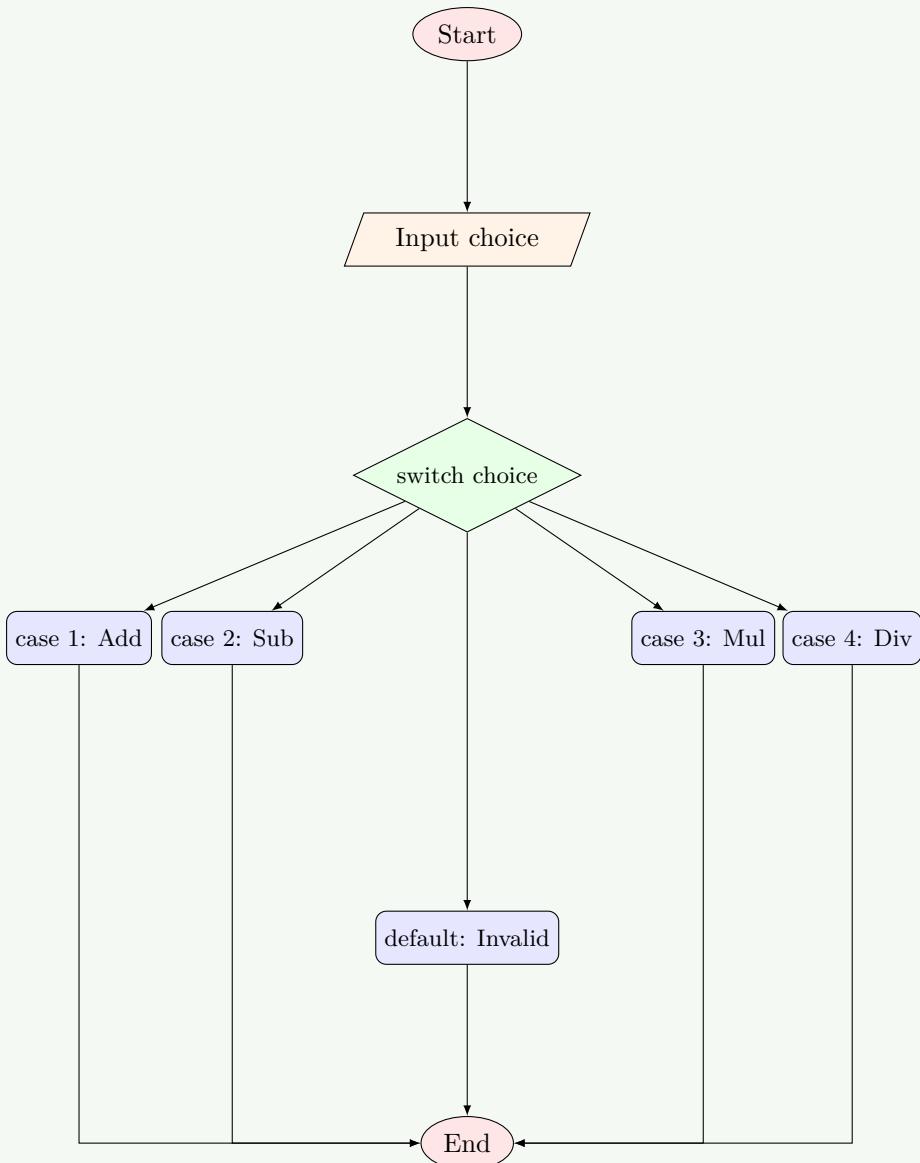
### Question 3 [c marks]

7 Draw flowchart and explain switch statement with example.

#### Solution

##### Answer:

The switch statement selects one code block from multiple options based on a variable's value.



```

1 #include <stdio.h>
2
3 int main() {
  
```

```

4     int choice;
5
6     printf("Enter operation (1-4): ");
7     scanf("%d", &choice);
8
9     switch(choice) {
10        case 1:
11            printf("Addition selected");
12            break;
13        case 2:
14            printf("Subtraction selected");
15            break;
16        case 3:
17            printf("Multiplication selected");
18            break;
19        case 4:
20            printf("Division selected");
21            break;
22        default:
23            printf("Invalid choice");
24    }
25
26    return 0;
27 }
```

- **Expression:** Takes integer or character expression
- **Case labels:** Must be constant expressions
- **Break statement:** Prevents fall-through to next case
- **Default:** Handles values not matching any case

### Mnemonic

”SCBD - Switch Cases Break Default”

## Question 4 [a marks]

3 Develop a C program to convert uppercase alphabet to lowercase alphabet.

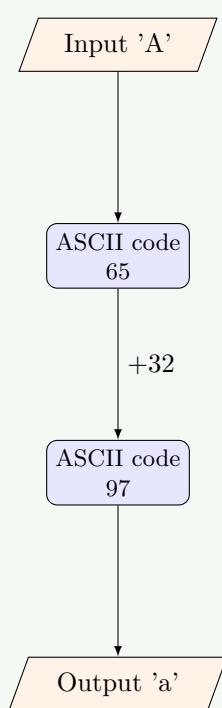
### Solution

#### Answer:

```

1 #include <stdio.h>
2
3 int main() {
4     char upper, lower;
5
6     printf("Enter uppercase letter: ");
7     scanf("%c", &upper);
8
9     lower = upper + 32;
10    // Alternatively: lower = tolower(upper);
11
12    printf("Lowercase letter: %c", lower);
13
14    return 0;
15 }
```

#### Diagram:

**Mnemonic**

"ASCII-32 - Add 32 to Shift Characters Into Lowercase"

## Question 4 [b marks]

4 What is pointer? Explain with example.

### Solution

**Answer:**

A pointer is a variable that stores the memory address of another variable.

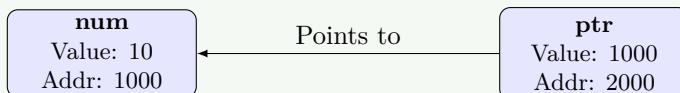
Concept	Syntax	Description
Declaration	int *p;	Declares pointer p to int
Initialization	p = &var;	Store address of var in p
Dereferencing	*p = 10;	Access/modify pointed value
Pointer arithmetic	p++	Move to next memory location

```

1 #include <stdio.h>
2
3 int main() {
4     int num = 10;
5     int *ptr;
6
7     ptr = &num; // Store address of num in ptr
8
9     printf("Value of num: %d\n", num);
10    printf("Address of num: %p\n", &num);
11    printf("Value of ptr: %p\n", ptr);
12    printf("Value pointed by ptr: %d\n", *ptr);
13
14    *ptr = 20; // Change value using pointer
15    printf("New value of num: %d\n", num);
  
```

```

16
17     return 0;
18 }
```

**Memory Diagram:****Mnemonic**

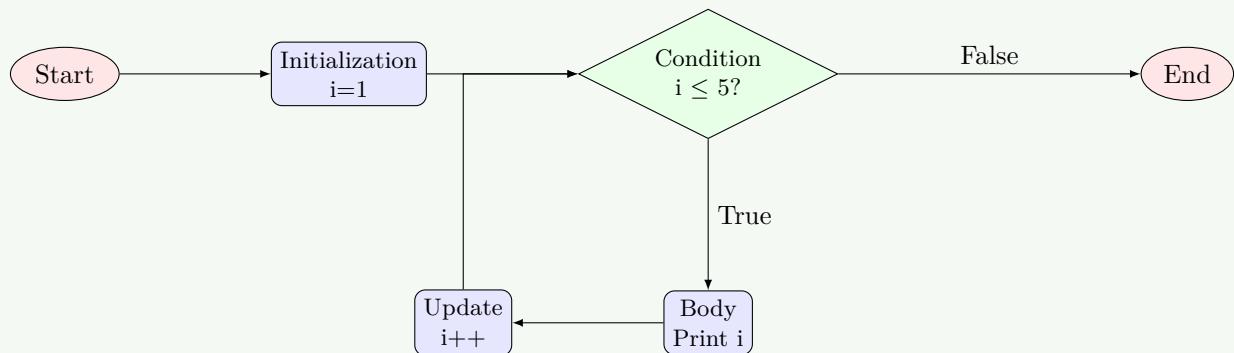
"SAID - Store Address to Indirectly Dereference"

**Question 4 [c marks]**

7 Draw flowchart and explain for loop with example.

**Solution****Answer:**

The for loop is used to repeat a block of code a specified number of times.



```

1 #include <stdio.h>
2
3 int main() {
4     int i;
5
6     // Syntax: for(initialization; condition; update)
7     for(i = 1; i <= 5; i++) {
8         printf("%d ", i);
9     }
10    // Output: 1 2 3 4 5
11
12    return 0;
13 }
```

- **Initialization:** Executes once before loop starts
- **Condition:** Checked before each iteration
- **Update:** Executes after each iteration
- **Body:** Code block that repeats

**Mnemonic**

"ICU-B - Initialize, Check, Update, Body"

OR

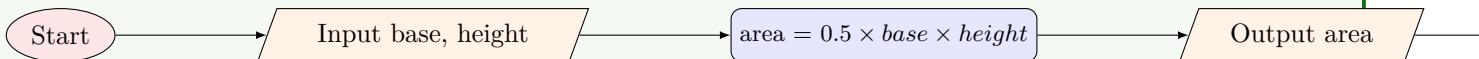
**Question 4 [a marks]**

- 3 Develop a C program to find area of a triangle ( $0.5 * \text{base} * \text{height}$ )?

**Solution****Answer:**

```

1 #include <stdio.h>
2
3 int main() {
4     float base, height, area;
5
6     printf("Enter base of triangle: ");
7     scanf("%f", &base);
8
9     printf("Enter height of triangle: ");
10    scanf("%f", &height);
11
12    area = 0.5 * base * height;
13
14    printf("Area of triangle: %.2f", area);
15
16    return 0;
17 }
```

**Diagram:****Mnemonic**

"BHA - Base times Height divided by two equals Area"

OR

**Question 4 [b marks]**

- 4 Explain declaration and initialization of pointer.

**Solution****Answer:**

Operation	Syntax	Example	Description
Declaration	datatype *name;	int *ptr;	Creates a pointer variable
Initialization	name = &var;	ptr = &num;	Assigns address to pointer
Combined	datatype *name = &var;	int *ptr = &num;	Declaration with initialization
NULL pointer	name = NULL;	ptr = NULL;	Safe initialization

```

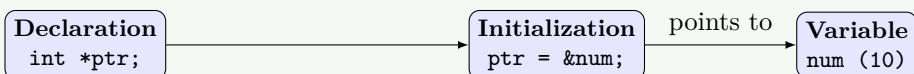
1 #include <stdio.h>
2
3 int main() {
4     int num = 10;           // Regular variable

```

```

5     int *ptr1;           // Declaration only
6     int *ptr2 = &num;      // Declaration with initialization
7
8     ptr1 = &num;          // Initialization of ptr1
9
10    printf("num value: %d\n", num);
11    printf("num address: %p\n", &num);
12    printf("ptr1 value: %p\n", ptr1);
13    printf("ptr2 value: %p\n", ptr2);
14
15    return 0;
16 }
```

Diagram:



### Mnemonic

"PAIN - Pointer Allocate, Initialize, Navigate"

OR

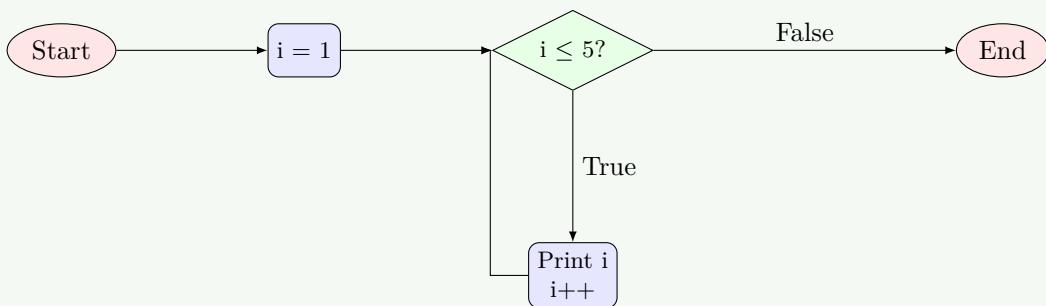
## Question 4 [c marks]

7 Draw flowchart and explain while loop with example.

### Solution

#### Answer:

The while loop repeats a block of code as long as a specified condition is true.



```

1 #include <stdio.h>
2
3 int main() {
4     int i = 1;
5
6     // Syntax: while(condition)
7     while(i <= 5) {
8         printf("%d ", i);
9         i++;
10    }
11    // Output: 1 2 3 4 5
12
13    return 0;
14 }
```

- **Entry controlled:** Condition checked before execution
- **Infinite loop:** If condition never becomes false
- **Components:** Initialization, Condition, Body, Update

**Mnemonic**

"WET - While Entry Test"

## Question 5 [a marks]

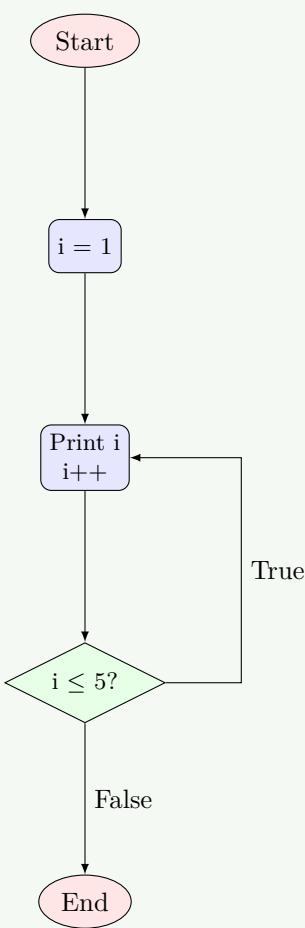
3 Explain do...while loop with example.

**Solution****Answer:**

The do-while loop is an exit-controlled loop that executes the block of code at least once.

```
1 #include <stdio.h>
2
3 int main() {
4     int i = 1;
5
6     do {
7         printf("%d ", i);
8         i++;
9     } while(i <= 5);
10
11    return 0;
12 }
```

Flowchart:

**Mnemonic**

"DEP - Do Exit Post-test"

**Question 5 [b marks]**

4 Write a program to input 5 numbers in array and display it.

**Solution****Answer:**

```

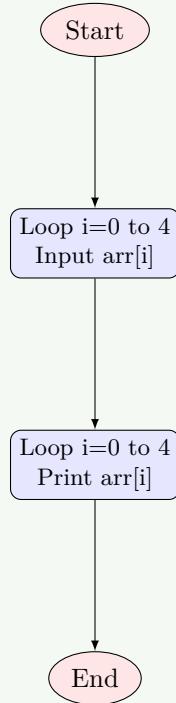
1 #include <stdio.h>
2
3 int main() {
4     int arr[5], i;
5
6     printf("Enter 5 numbers:\n");
7     for(i = 0; i < 5; i++) {
8         scanf("%d", &arr[i]);
9     }
10
11    printf("Array elements are:\n");
12    for(i = 0; i < 5; i++) {
13        printf("%d ", arr[i]);
14    }
15

```

```

16     return 0;
17 }
```

Diagram:



#### Mnemonic

"LIO - Loop Input Output"

## Question 5 [c marks]

7 Explain User Defined Function (UDF) with example. categories of UDF.

#### Solution

##### Answer:

A User Defined Function is a block of code created by the user to perform a specific task.

##### Components:

- Function Declaration (Prototype)
- Function Definition (Body)
- Function Call

##### Categories of UDF:

Category	Description
1. No arguments, No return value	Simplest form, just executes code
2. With arguments, No return value	Takes input, returns nothing
3. No arguments, With return value	Takes no input, returns result
4. With arguments, With return value	Takes input, returns result

##### Example (Cat 4):

```

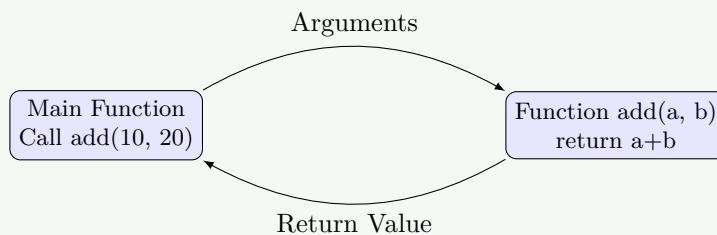
1 #include <stdio.h>
2
3 // 1. Declaration
```

```

4 int add(int a, int b);
5
6 int main() {
7     int sum;
8     // 3. Call
9     sum = add(10, 20);
10    printf("Sum: %d", sum);
11    return 0;
12 }
13
14 // 2. Definition
15 int add(int a, int b) {
16     return a + b;
17 }

```

Diagram:



### Mnemonic

"DDC - Declare Define Call"

OR

## Question 5 [a marks]

3 What is structure? Explain with example.

### Solution

#### Answer:

Structure is a user-defined data type available in C that allows to combine data items of different kinds.

```

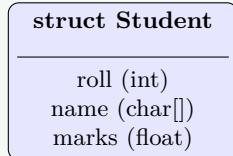
1 #include <stdio.h>
2 #include <string.h>
3
4 struct Student {
5     int roll;
6     char name[20];
7     float marks;
8 };
9
10 int main() {
11     struct Student s1;
12
13     s1.roll = 1;
14     strcpy(s1.name, "Raju");
15     s1.marks = 85.5;
16
17     printf("Roll: %d\n", s1.roll);
18     printf("Name: %s\n", s1.name);

```

```

19     printf("Marks: %.2f\n", s1.marks);
20
21     return 0;
22 }
```

Diagram:



### Mnemonic

"GROUP - Group Related Objects Under Pattern"

OR

## Question 5 [b marks]

4 Explain any four string handling functions.

### Solution

#### Answer:

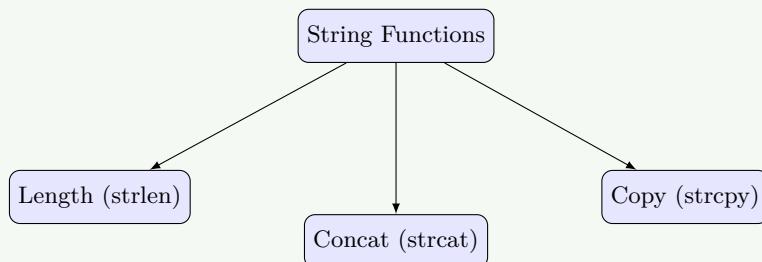
Function	Description	Example
strlen(s)	Returns length of string	strlen("Hello") = 5
strcpy(d, s)	Copies source to dest	strcpy(s1, "Hi")
strcat(d, s)	Concatenates s to d	strcat("Hi", "There")
strcmp(s1, s2)	Compares two strings	strcmp("A", "B") = -1

#### Example:

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main() {
5     char s1[20] = "Hello";
6     char s2[20] = "World";
7
8     printf("Length: %d\n", strlen(s1));
9     strcat(s1, s2);
10    printf("Concat: %s\n", s1);
11
12    return 0;
13 }
```

Diagram:



**Mnemonic**

"LCCC - Len Copy Cat Cmp"

OR

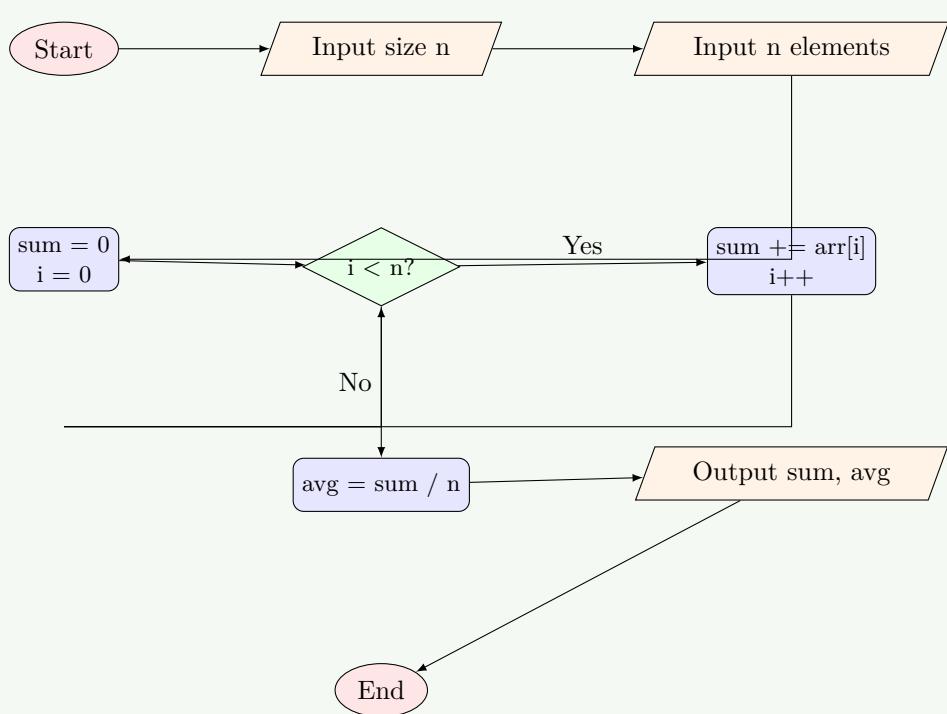
**Question 5 [c marks]**

7 Develop a C program to find sum of array elements and average of it.

**Solution****Answer:**

```
1 #include <stdio.h>
2
3 int main() {
4     int arr[100], n, i;
5     int sum = 0;
6     float avg;
7
8     printf("Enter number of elements: ");
9     scanf("%d", &n);
10
11    printf("Enter %d elements:\n", n);
12    for(i = 0; i < n; i++) {
13        scanf("%d", &arr[i]);
14        sum += arr[i]; // Add each element to sum
15    }
16
17    avg = (float)sum / n; // Calculate average
18
19    printf("Sum of array elements: %d\n", sum);
20    printf("Average of array elements: %.2f", avg);
21
22    return 0;
23 }
```

**Diagram:**



Step	Operation	Example (for [5,10,15,20])
1	Input array	[5,10,15,20]
2	Initialize sum = 0	sum = 0
3	Add each element	sum = 50
4	Divide by count	avg = 12.5

**Mnemonic**

"LISA - Loop, Increment, Sum, Average"