

# Subject Name (Gujarati)

4331105 -- Winter 2022

Semester 1 Study Material

*Detailed Solutions and Explanations*

## પ્રશ્ન 1(અ) [3 ગુણ]

C લોગેજના બેઝીક ડેટા ટાઈપ તેની રેજ સાથે લખો

### જવાબ

ડેટા ટાઈપ	સાઈઝ (બાઇટ્સ)	રેજ
char	1	-128 થી 127
int	2 અથવા 4	-32,768 થી 32,767 (2 બાઇટ્સ) અથવા -2,147,483,648 થી 2,147,483,647 (4 બાઇટ્સ)
float	4	3.4E-38 થી 3.4E+38
double	8	1.7E-308 થી 1.7E+308

### મેમરી ટ્રીક

“CIFD - Computer Is Fundamentally Digital”

## પ્રશ્ન 1(બ) [4 ગુણ]

વેરીએબલના નામ માટેના નિયમો સમજાવો.

### જવાબ

નિયમ	ઉદાહરણ
અક્ષર અથવા અંડરસ્કોરથી શરૂ થતું જોઈએ	માન્ય: _count, name / અમાન્ય: 1score
અક્ષરો, અંકો, અંડરસ્કોર ધરાવી શકે	માન્ય: user_1 / અમાન્ય: user-1
કીવર્ડ્સનો ઉપયોગ ન થઈ શકે	માન્ય: integer / અમાન્ય: int
કેસ સેન્સિટિવ	total અને TOTAL અલગ છે

### આફ્ટિટ:

Variable Naming Rules

[A{-Z}, a{-}z, \\_] [A{-}Z, a{-}z, 0{-}9, \\_]\* }

### મેમરી ટ્રીક

“LUCK - Letters Underscore Case Keywords”

## પ્રશ્ન 1(ક) [7 ગુણ]

ફ્લોચાર્ટની વ્યાખ્યા લખો. બે પૂર્ણાંક નંબર N1 અને N2 માંથી નાનો નંબર શોધવા માટેનો ફ્લોચાર્ટ દોરો.

## જવાબ

ફ્લોચાર્ટ એ એલોરિધમનું ગ્રાફિકલ રજૂઆત છે જે પ્રમાણિત સિમ્બોલ્સનો ઉપયોગ કરીને એરો દ્વારા જોડાયેલા પગલાંઓની કમને દર્શાવે છે.

```
flowchart LR
    A([Start]) --> B[/Input N1, N2/]
    B --> C{N1 N2?}
    C -- Yes --> D[min = N1]
    C -- No --> E[min = N2]
    D --> F[/Output min/]
    E --> F
    F --> G([End])
```

- ફ્લોચાર્ટ સિમ્બોલ્સ: તાઇક પગલાંઓની દૃશ્ય રજૂઆત
- નિર્ણય ડાયમ્બ: શરત તપાસીને વહેણ પથ નક્કી કરે છે
- પ્રોસેસ બોક્સોસ: ગણતરી અથવા ઓપરેશન્સ ઘરાવે છે

## મેમરી ટ્રીક

“FAST - Flow Analysis Shown Through-charts”

## પ્રશ્ન 1(ક OR) [7 ગુણ]

અલગોરિથમની વ્યાખ્યા લખો. વર્તુળનું ક્ષેત્રફળ અને પરિધ ગણવા માટેનો અલગોરિથમ લખો.

## જવાબ

અલગોરિથમ એ કોઈ ચોક્કસ સમસ્યાને ઉકેલવા માટેની સ્પષ્ટ રીતે વ્યાખ્યાયિત સૂચનાઓની પરિમિત કમમાં પગલે-પગલાની પ્રક્રિયા છે.  
વર્તુળની ગણતરી માટેનો અલગોરિથમ:

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

પગલું	ઓપરેશન	ફોર્મ્યુલા
1	ત્રિજ્યા મેળવો	Input r
2	ક્ષેત્રફળની ગણતરી	$A = \pi \times r^2$
3	પરિધની ગણતરી	$C = 2 \times \pi \times r$
4	પરિણામ દર્શાવો	Output A, C

## મેમરી ટ્રીક

“SICS - Steps In Clear Sequence”

## પ્રશ્ન 2(અ) [3 ગુણ]

printf() અને scanf() વર્ણનો તફાવત સમજાવો.

## જવાબ

ફીચર	printf()	scanf()
ઉદ્દેશ	સ્ક્રીન પર ડેટા આઉટપુટ કરે છે	ક્રીબોક્સ ડેટા ઇનપુટ લે છે
ફોર્મેટ	printf("format", variables)	scanf("format", &variables)
રિટર્ન	છાપેલા અક્ષરોની સંખ્યા	સફળતાપૂર્વક વાંચેલી આઇટમોની સંખ્યા

## મેમરી ટ્રીક

"IO-AR - Input Output-Address Returns"

## પ્રશ્ન 2(બ) [4 ગુણ]

કન્ડીશનલ ઓપરેટરનો ઉપયોગ કરીને વે નંબરમાંથી મોટો નંબર શોધવા માટે C પ્રોગ્રામ બનાવો.

## જવાબ

```
\#include <stdio.h>

int main() {
    int num1, num2, max;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    max = (num1 > num2) ? num1 : num2;

    printf("Maximum number is: %d", max);

    return 0;
}
```

## આફ્ટિ:

Input	{ Condition	Output	}
num1, num2	num1 > num2?	max	}

## મેમરી ટ્રીક

"CTO - Condition Then Output"

## પ્રશ્ન 2(ક) [7 ગુણ]

એરીથમેટીક અને રીલેશનલ ઓપરેટરો ઉદાહરણ સાથે સમજાવો.

## જવાબ

પ્રકાર	ઓપરેટર્સ	ઉદાહરણ	પરિણામ
<b>એરીથમેટીક ઓપરેટર્સ</b>			
સરવાળો	+	5 + 3	8
બાદબાકી	-	5 - 3	2
ગુણાકાર	*	5 * 3	15
ભાગાકાર	/	5 / 3	1 (પૂર્ણક ભાગાકાર)
મોડ્યુલસ	%	5 % 3	2 (રોષ)
<b>રીલેશનલ ઓપરેટર્સ</b>			
સમાન	==	5 == 3	0 (ખોટું)
અસમાન	!=	5 != 3	1 (સાચું)
મોટું	>	5 > 3	1 (સાચું)
નાનું	<	5 < 3	0 (ખોટું)

મોટું અથવા સમાન	$>=$	$5 >= 5$	1 (સાચું)
નાનું અથવા સમાન	$<=$	$5 <= 3$	0 (ખોટું)

### મેમરી ટ્રીક

“ASMDCRO - Add Subtract Multiply Divide Compare Return Output”

### પ્રશ્ન 2(અ OR) [3 ગુણ]

જો  $(25/3) * 4 - 10 \% 3 + 9/2$  સમીકરણને ઉકેલવામાં આવે તો ઓપરેટોની અગ્રીમતાને ધ્યાને લઈ દરેક સ્ટેપ અને અંતિમ જવાબ લખો.

#### જવાબ

પગલું	ઓપરેશન	ગણતરી	પરિણામ
1	કૌસ (25/3)	$25/3 = 8$ (પૂર્ણક ભાગાકાર)	8
2	મોડ્યુલસ 10 % 3	$10 \% 3 = 1$	1
3	ભાગાકાર 9/2	$9/2 = 4$ (પૂર્ણક ભાગાકાર)	4
4	ગુણાકાર $8 * 4$	$8 * 4 = 32$	32
5	બાદબાકી $32 - 1$	$32 - 1 = 31$	31
6	સરવાળો $31 + 4$	$31 + 4 = 35$	35

અંતિમ જવાબ = 35

### મેમરી ટ્રીક

“PEMDAS - Parentheses, Exponents, Multiplication/Division, Addition/Subtraction”

### પ્રશ્ન 2(બ OR) [4 ગુણ]

એલ્જેબ્રિક સમીકરણના ઉકેલ માટેનો C પ્રોગ્રામ લખો.

#### જવાબ

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

int main() {
    float a, b, c;
    float discriminant, root1, root2;

    printf("Enter coefficients a, b, c: ");
    scanf("%f %f %f", &a, &b, &c);

    discriminant = b*b - 4*a*c;

    if (discriminant < 0) {
        root1 = (-b + sqrt(discriminant)) / (2*a);
        root2 = (-b - sqrt(discriminant)) / (2*a);
        printf("Roots: %.2f and %.2f", root1, root2);
    } else if (discriminant == 0) {
        root1 = -b / (2*a);
        printf("Root: %.2f", root1);
    } else {
        printf("No real roots");
    }

    return 0;
}
```

## આફ્ટિં:

```

flowchart LR
    A[Input a,b,c] --> B[Calculate d = b^2 - 4ac]
    B --> C{d == 0?}
    C -- Yes --> D[Two real roots]
    C -- No --> E{d == 0?}
    E -- Yes --> F[One real root]
    E -- No --> G[No real roots]

```

## મેમરી ટ્રીક

“QDR - Quadratic Discriminant Roots”

## પ્રશ્ન 2(ક OR) [7 ગુણ]

લોજિકલ અને બીટ-વાઈઝ ઓપરેટરો ઉદાહરણ સાથે સમજાવો.

### જવાબ

પ્રકાર	ઓપરેટર્સ	ઉદાહરણ	પરિણામ
<b>લોજિકલ ઓપરેટર્સ</b>			
લોજિકલ AND	&&	(5>3) && (4<7)	1 (સાચું)
લોજિકલ OR		(5<3)    (4<7)	1 (સાચું)
લોજિકલ NOT	!	!(5>3)	0 (ખોટું)
<b>બિટવાઈઝ ઓપરેટર્સ</b>			
બિટવાઈઝ AND	&	5 & 3 (101 & 011)	1 (001)
બિટવાઈઝ OR		5   3 (101   011)	7 (111)
બિટવાઈઝ XOR	^	5 ^ 3 (101 ^ 011)	6 (110)
બિટવાઈઝ NOT	~	~5 (~ 00000101)	-6 (11111010)
લેફ્ટ શિફ્ટ	<<	5 << 1 (101 << 1)	10 (1010)
રાઇટ શિફ્ટ	>>	5 >> 1 (101 >> 1)	2 (10)

## મેમરી ટ્રીક

“LAND BORNS - Logical AND OR NOT, Bitwise OR AND NOT Shift”

## પ્રશ્ન 3(અ) [3 ગુણ]

‘go to’ સ્ટેટમેન્ટનો ઉપયોગ ઉદાહરણ સાથે સમજાવો.

### જવાબ

goto સ્ટેટમેન્ટ પ્રોગ્રામમાં લેબલ કરેલા સ્ટેટમેન્ટ પર બિનશરતી જમ્પની મંજૂરી આપે છે.

```

#include <stdio.h>

int main() {
    int i = 0;

start:
    printf("\%d ", i);
    i++;
    if (i == 5)
        goto start;

    return 0;
}

```

```
\}
// : 0 1 2 3 4
```

આફ્રતિ:

Start

```
print(i)
i++
```

Yes  
i { 5? }

No

End

```
goto start
```

### મેમરી ટ્રીક

“JUMP - Just Unconditionally Move Program-counter”

### પ્રશ્ન 3(બ) [4 ગુણ]

લીધેલ નંબર એકી છે કે બેકી તે તપાસવા માટેનો C પ્રોગ્રામ લખો.

#### જવાબ

```
\#include <stdio.h>

int main() {
    int num;

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

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

    return 0;
}
```

આફ્રતિ:

```
flowchart LR
    A([Start]) --> B[/Input num/]
    B --> C{num \% 2 == 0?}
    C -- Yes --> D[/Print num is even/]
```

```

C {-{-}|No| E[/Print num is odd/]}
D {-{-} F([End])}
E {-{-} F}

```

## મેમરી ટ્રીક

“MODE - Modulo Odd-Even Determination”

### પ્રશ્ન 3(ક) [7 ગુણ]

else if ladder ફ્લોચાર્ટ દોરી ઉદાહરણ સાથે સમજાવો.

#### જવાબ

else-if લેડર કધામ શરતોને કમમાં ચકાસવાની મંજૂરી આપે છે, પ્રથમ સાચી શરત સાથે જોડાયેલા બ્લોકને એક્ઝિક્યુટ કરે છે.

```

flowchart LR
A([Start]) --> B[/Input marks/]
B --> C{marks = 90?}
C -- Yes --> D[grade = A]
C -- No --> E{marks = 80?}
E -- Yes --> F[grade = B]
E -- No --> G{marks = 70?}
G -- Yes --> H[grade = C]
G -- No --> I{marks = 60?}
I -- Yes --> J[grade = D]
I -- No --> K[grade = F]
D & F & H & J & K --> L[/Output grade/]
L --> M([End])

```

```

#include <stdio.h>

int main() {
    int marks;
    char grade;

    printf("Enter marks: ");
    scanf("%d", &marks);

    if (marks == 90)
        grade = {A};
    else if (marks == 80)
        grade = {B};
    else if (marks == 70)
        grade = {C};
    else if (marks == 60)
        grade = {D};
    else
        grade = {F};

    printf("Grade: %c", grade);

    return 0;
}

```

- એકાધિક શરતો: કમાનુસાર શરતો તપાસે છે
- પ્રથમ મિલાન: માત્ર પ્રથમ સાચી શરત માટે કોડ એક્ઝિક્યુટ કરે છે
- ડિફોલ્ટ કેસ: છેલ્લો else બાકીના બધા કેસને સંબાને છે

## પ્રશ્ન 3(અ OR) [3 ગુણ]

continue અને break સ્ટેટમેન્ટનો ઉપયોગ સમજાવો.

## જવાબ

સ્ટેટમેન્ટ	હેતુ	અસર
break	લૂપ અથવા સિવચમાંથી બહાર નીકળવું	તરત જ સમગ્ર લૂપ છોડી દે છે
continue	વર્તમાન પુનરાવર્તન છોડવું	લૂપના આગલા પુનરાવર્તન પર જાય છે

```
// break
for(int i=1; i{=}10; i++) \{

    if(i == 6)
        break;      // i=6
    printf("\%d ", i); //      : 1 2 3 4 5
\}

// continue
for(int i=1; i{=}10; i++) \{

    if(i \% 2 == 0)
        continue;  //
    printf("\%d ", i); //      : 1 3 5 7 9
\}
```

## આફ્ટેટ:

break	continue
Loop	Loop
1	1
2	2
3	break
	continue
3	3
Exit Loop	Next Iteration

## પ્રશ્ન 3(બ OR) [4 ગુણ]

૧ થી ૧૦ નંબરનો સરવાળો પ્રિન્ટ કરવા માટેનો C પ્રોગ્રામ ફોર લૂપનો ઉપયોગ કરીને લખો.

## જવાબ

```
\#include <stdio.h>

int main() {
    int i, sum = 0;

    for(i = 1; i <= 10; i++) {
        sum += i;
    }

    printf("Sum of numbers from 1 to 10: %d", sum);

    return 0;
}
```

## આફ્ટરિંગ:

```
flowchart LR
    A([Start]) --> B[B[sum = 0]]
    B --> C[C[i = 1]]
    C --> D{D{i = 10?}}
    D -- Yes --> E[E[sum = sum + i]]
    E --> F[F[i++]]
    F --> D
    D -- No --> G[G[/Print sum/]]
    G --> H([End])
```

## મેમરી ટ્રીક

``SILA - Sum Increment Loop Add''

## પ્રશ્ન 3(ક) OR [7 ગુણ]

સ્વીચ સ્ટેટમેન્ટ ફ્લોચાર્ટ દ્વારી ઉદાહરણ સાથે સમજાવો.

## જવાબ

સ્વીચ સ્ટેટમેન્ટ વેરિએબલની વેલ્યુના આધારે એકાધિક વિકલ્પોમાંથી એક કોડ બલોક પસંદ કરે છે.

```
flowchart LR
    A([Start]) --> B[/Input choice/]
    B --> C{\switch choice}
    C -- case 1 --> D[/Print Addition/]
    C -- case 2 --> E[/Print Subtraction/]
    C -- case 3 --> F[/Print Multiplication/]
    C -- case 4 --> G[/Print Division/]
    C -- default --> H[/Print Invalid/]
    D & E & F & G & H --> I([End])
```

```
\#include <stdio.h>

int main() {
    int choice;

    printf("Enter operation (1-4): ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("Addition selected");
            break;
```

```

        case 2:
            printf("Subtraction selected");
            break;
        case 3:
            printf("Multiplication selected");
            break;
        case 4:
            printf("Division selected");
            break;
        default:
            printf("Invalid choice");
    }

    return 0;
}

```

- એક્સપ્રેશન: ઇન્ટિજર અથવા કોરેક્ટર એક્સપ્રેશન લે છે
- કેસ લિબલ્સ: કોન્સ્ટન્ટ એક્સપ્રેશન હોવા જોઈએ
- બેક સ્ટેમેન્ટ: આગલા કેસમાં પ્રવેશ થતો અટકાવે છે
- ડિફોલ્ટ: કોઈપણ કેસ મેચ ન થાય તો હેન્ડલ કરે છે

## મેમરી ટ્રીક

“SCBD - Switch Cases Break Default”

### પ્રશ્ન 4(અ) [૩ ગુણ]

મોટા કેપિટલ મૂળાક્ષરોને નાના મૂળાક્ષરોમાં રૂપાંતર કરવા માટેનો C પ્રોગ્રામ લખો.

#### જવાબ

```

#include <stdio.h>

int main() {
    char upper, lower;

    printf("Enter uppercase letter: ");
    scanf("\%c", &upper);

    lower = upper + 32;
    //      : lower = tolower(upper);

    printf("Lowercase letter: \%c", lower);

    return 0;
}

```

#### આફ્ટરિટિ:

Input {A }

ASCII code  
65

+32

ASCII code

```
Output {a }
```

### મેમરી ટ્રીક

“ASCII-32 - Add 32 to Shift Characters Into Lowercase”

### પ્રશ્ન 4(બ) [4 ગુણ]

પોઇન્ટર એટલે શું? ઉદાહરણ સાથે સમજાવો.

#### જવાબ

પોઇન્ટર એ એક વેરિએબલ છે જે અન્ય વેરિએબલના મેમરી એડ્રેસને સ્ટોર કરે છે.

કોન્સેપ્ટ	સિન્ટેક્સ	વિગત
ડેકલરેશન	int *p;	ઇન્પ્રોક્સનો પોઇન્ટર p ડિક્લેર કરે છે
ઇનિશિયલાઇઝેશન	p = &var;	var નું એડ્રેસ p માં સ્ટોર કરે છે
ડિરેફરન્સિંગ	*p = 10;	પોઇન્ટ થયેલી વેલ્યુને એક્સેસ/મોડિફિય કરે છે
પોઇન્ટર એરિથમેટિક	p++	આગલા મેમરી લોકેશન પર જાય છે

```
\#include <stdio.h>

int main() {
    int num = 10;
    int *ptr;

    ptr = &num; // num      ptr

    printf("Value of num: %d\n", num);
    printf("Address of num: %p\n", &num);
    printf("Value of ptr: %p\n", ptr);
    printf("Value pointed by ptr: %d\n", *ptr);

    *ptr = 20; // New value of num: %d
    printf("New value of num: %d\n", num);

    return 0;
}
```

### આફ્ટિ:

Memory:

num	1000
(10)	*ptr
	2000
	(1000)

### મેમરી ટ્રીક

“SAID - Store Address to Indirectly Dereference”

## પ્રશ્ન 4(ક) [7 ગુણ]

ફોર લૂપ ફ્લોચાર્ટ દોરી ઉદાહરણ સાથે સમજાવો.

### જવાબ

ફોર લૂપનો ઉપયોગ કોડના બ્લોકને નિર્દિષ્ટ સંઘામાં વાર રિપીટ કરવા માટે થાય છે.

```

flowchart LR
    A([Start]) --> B[Initialization: i=1]
    B --> C{Condition: i=5?}
    C -- True --> D[Body: Print i]
    D --> E[Update: i++]
    E --> C
    C -- False --> F([End])

```

```

#include <stdio.h>

int main() {
    int i;

    // : for(initialization; condition; update)
    for(i = 1; i != 5; i++) {
        printf("\%d ", i);
    }
    // : 1 2 3 4 5

    return 0;
}

```

- ઇનિશિયલાઇઝેશન: લૂપ શરૂ થતાં પહેલાં એક વાર એક્ઝિક્યુટ થાય છે
- કન્ડિશન: દરેક પુનરાવર્તન પહેલાં ચેક થાય છે
- અપડેટ: દરેક પુનરાવર્તન પછી એક્ઝિક્યુટ થાય છે
- બોડી: કોડ બ્લોક જે રિપીટ થાય છે

### મેમરી ટ્રીક

“ICU-B - Initialize, Check, Update, Body”

## પ્રશ્ન 4(અ OR) [3 ગુણ]

ત્રિકોણનું ક્ષેત્રફળ ( $1/2 * પાયો * ઊંચાઈ$ ) શોધવા માટેનો C પ્રોગ્રામ લખો.

### જવાબ

```

#include <stdio.h>

int main() {
    float base, height, area;

    printf("Enter base of triangle: ");
    scanf("\%f", &base);

    printf("Enter height of triangle: ");
    scanf("\%f", &height);

    area = 0.5 * base * height;

    printf("Area of triangle: \%.2f", area);

    return 0;
}

```

આફ્ટિં:

```
flowchart LR
    A([Start]) --> B[/Input base, height/]
    B --> C[area = 0.5 * base * height]
    C --> D[/Output area/]
    D --> E([End])
```

મેમરી ટ્રીક

“BHA - Base times Height divided by two equals Area”

#### પ્રશ્ન 4(બ OR) [4 ગુણ]

પોઇન્ટરનું ડિક્લેરેશન અને ઈન્નીશ્યલાઈજેશન સમજાવો.

જવાબ

ઓપરેશન	સિન્ટેક્સ	ઉદાહરણ	વિગત
ડેક્લેરેશન	datatype *pointer_name;	int *ptr;	પોઇન્ટર વેરિએબલ બનાવે છે
ઇનિશિયલાઈજેશન	pointer_name = &variable;	ptr = &num;	પોઇન્ટરને એડ્રેસ સોંપે છે
કોમ્બાઇન	datatype *pointer_name = &variable;	int *ptr = &num;	ડેક્લેરેશન સાથે ઇનિશિયલાઈજેશન
NULL પોઇન્ટર	pointer_name = NULL;	ptr = NULL;	જ્યારે એડ્રેસ ઉપલબ્ધ ન હોય ત્યારે સુરક્ષિત ઇનિશિયલાઈજેશન

```

\#include <stdio.h>

int main() {
    int num = 10;           //
    int *ptr1;             //
    int *ptr2 = &num;       //

    ptr1 = &num;           // ptr1

    printf("num value: %d\n", num);
    printf("num address: %p\n", &num);
    printf("ptr1 value: %p\n", ptr1);
    printf("ptr2 value: %p\n", ptr2);
    printf("Value via ptr1: %d\n", *ptr1);
    printf("Value via ptr2: %d\n", *ptr2);

    return 0;
}

```

**આકૃતિ:**

Pointer Declaration:

```
int *ptr;
```

Pointer Initialization:

```
ptr = &num;           num (10)
```

## મેમરી ટ્રીક

“PAIN - Pointer Allocate, Initialize, Navigate”

## પ્રશ્ન 4(ક OR) [7 ગુણ]

વાઇલ લૂપ ફ્લોચાર્ટ દોરી ઉદાહરણ સાથે સમજાવો.

### જવાબ

વાઇલ લૂપ એક સ્પષ્ટ કરેલી શરત જ્યાં સુધી સાચી હોય ત્યાં સુધી કોડના બ્લોકને રિપીટ કરે છે.

```

flowchart LR
    A([Start]) --> B[i = 1]
    B --> C{i = 5?}
    C -- True --> D[/Print i/]
    D --> E[i++]
    E --> C
    C -- False --> F([End])

\#include <stdio.h>

int main() {
    int i = 1;

    // : while(condition) { body }
    while(i != 5) {
        printf("%d ", i);
        i++;
    }
    // : 1 2 3 4 5
}

```

```

    return 0;
}

• ઇનિશિયલાઇઝન: લૂપ પહેલાં કરવું જરૂરી છે
• કન્ડિશન: દરેક પુનરાવર્તનની શરૂઆતમાં મૂલ્યાંકન થાય છે
• બોડી: માત્ર જો શરત સાચી હોય તો જ એક્ઝિક્યુટ થાય છે
• અપડેટ: લૂપ બોડીની અંદર હોવું જોઈએ

```

## મેમરી ટ્રીક

``CUBE - Condition check, Update inside Body, Exit when false''

## પ્રશ્ન 5(અ) [3 ગુણ]

બુકની આપેલ માહિતી સ્ટોર કરવાનું સ્ટ્રક્ચર બનાવો: book\_no, book\_title, book\_author, book\_price

### જવાબ

```

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

struct Book {
    int book_no;
    char book_title[50];
    char book_author[50];
    float book_price;
};

int main() {
    struct Book book1;

    book1.book_no = 101;
    strcpy(book1.book_title, "Programming in C");
    strcpy(book1.book_author, "Dennis Ritchie");
    book1.book_price = 450.50;

    printf("Book No: %d\n", book1.book_no);
    printf("Title: %s\n", book1.book_title);
    printf("Author: %s\n", book1.book_author);
    printf("Price: %.2f", book1.book_price);

    return 0;
}

```

### આફ્ટર:

```

struct Book

book_no      (int)
book_title   (char[50])
book_author  (char[50])
book_price   (float)

```

## મેમરી ટ્રીક

``SNAP - Structure Needs All Properties''

## પ્રશ્ન 5(બ) [4 ગુણ]

(1)sqrt() (2)pow() (3)strlen() (4)strcpy() ફંક્શનો ઉદાહરણ સાથે સમજાવો.

### જવાબ

ફંક્શન	લાઇબ્રેરી	હેતુ	ઉદાહરણ
sqrt()	math.h	વર્ગમૂળ ગણો છે	sqrt(16) આઉટપુટ 4.0
pow()	math.h	પાવર ગણો છે	pow(2, 3) આઉટપુટ 8.0
strlen()	string.h	સ્ટ્રિંગની લંਬાઈ શોધે છે	strlen("hello") આઉટપુટ 5
strcpy()	string.h	સ્ટ્રિંગની કોપી કરે છે	strcpy(dest, "hello") dest માં "hello" કોપી કરે છે

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

int main() {
    double sqrtResult = sqrt(25);
    double powResult = pow(2, 4);
    char str[] = "Programming";
    char dest[20];
    int length = strlen(str);

    strcpy(dest, str);

    printf("sqrt(25) = %.2f\n", sqrtResult);
    printf("pow(2, 4) = %.2f\n", powResult);
    printf("Length of {}%s{} = {}%d\n", str, length);
    printf("Copied string: %s\n", dest);

    return 0;
}
```

### મેમરી ટ્રીક

"SPSS - Square-root Power String-length String-copy"

## પ્રશ્ન 5(ક) [7 ગુણ]

અરે અને અરેનું ઇનિશિયલાઇઝેશન ઉદાહરણ સાથે સમજાવો.

### જવાબ

અરે એ સમાન ડેટા એલિમેન્ટ્સનો સમૂહ છે જે સતત મેમરી લોકેશન પર સ્ટોર થાય છે.

પદ્ધતિ	સિન્ટેક્સ	ઉદાહરણ
ડેકલરેશન	data_type array_name[size];	int marks[5];
ડેકલરેશન સમયે ઇનિશિયલાઇઝેશન	data_type array_name[size] = {values};	int marks[5] = {95, 80, 85, 75, 90};
વક્તિગત એલિમેન્ટ	array_name[index] = value;	marks[0] = 95;
આંશિક ઇનિશિયલાઇઝેશન	int arr[5] = {1, 2};	બાકીના એલિમેન્ટ્સ 0 હોય છે
સાઇઝ વગર	int arr[] = {1, 2, 3};	સાઇઝ એલિમેન્ટ્સ દ્વારા નક્કી થાય છે

```

\#include <stdio.h>

int main() {
    //
    int numbers[5] = {10, 20, 30, 40, 50};

    //
    printf("First element: %d\n", numbers[0]);
    printf("Third element: %d\n", numbers[2]);

    //
    numbers[1] = 25;

    //
    printf("Array elements: ");
    for(int i = 0; i < 5; i++) {
        printf("%d ", numbers[i]);
    }
}

return 0;
}

```

**આજું:**

Array in memory:

10	20	30	40	50	
0	1	2	3	4	indices

### મેમરી ટ્રીક

“CASED - Contiguous Arrangement of Similar Elements with Direct-access”

### પ્રશ્ન 5(અ OR) [3 ગુણ]

એરે અને સ્ટ્રક્ચર નો તફાવત લખો.

#### જવાબ

ફીચર	એરે	સ્ટ્રક્ચર
ડેટા ટાઇપ્સ	માત્ર એક જ ડેટા ટાઇપ	અલગ અલગ ડેટા ટાઇપ્સ મંજૂર
એક્સેસ	ઇનેક્સ દ્વારા: arr[0]	ડોટ ઓપરેટર દ્વારા: emp.id
મેમરી	સતત ફુળવણી	સતત ન પણ હોઈ શકે
સાઈઝ	ડેક્લરેશન સમયે ફિક્સડ	સભ્યોના સાઈઝનો સરવાળો
ઇનિશિયલાઇઝેશન	int arr[3] = {1,2,3};	struct emp e = {101, "John", 5000};
હેતુ	સમાન આઇટમ્સનો સંગ્રહ	સંબંધિત આઇટમ્સનો સંગ્રહ

### આફ્ટિં:

Array:	Structure:
1 2 3	id: 101
int int int	name: "John"
	salary: 5000.0
	int char[] float

### મેમરી ટ્રીક

“HASDIP - Homogeneous vs. Assorted, Same vs. Different, Index vs. Point”

### પ્રશ્ન 5(બ) OR) [4 ગુણ]

યુઝર ડિફાઇન ફંક્શન એટલે શું? ઉદાહરણ સાથે સમજાવો.

#### જવાબ

યુઝર-ડિફાઇન ફંક્શન એ એક કોડ બ્લોક છે જે ચોક્કસ કાર્ય કરે છે, પ્રોગ્રામર દ્વારા કોઈને ફરીથી ઉપયોગ કરવા અને વ્યવસ્થિત કરવા માટે બનાવવામાં આવે છે.

કોમ્પોનેન્ટ	વિગત	ઉદાહરણ
રિટર્ન ટાઇપ	ફંક્શન દ્વારા પરત કરવામાં આવતા ડેટાનો પ્રકાર	int, void, વગેરે
ફંક્શન નામ	ફંક્શન માટેનું આઇડેન્ટિફિયર	sum, findMax
પેરામીટર્સ	કૌસમાં ઇનપુટ વેલ્યુઝ	(int a, int b)
ફંક્શન બોડી	કલ્યા બ્રેસિસ અંદરનો કોડ	{ return a+b; }

```
\#include <stdio.h>

// 
int sum(int a, int b);

int main() {
    int num1 = 5, num2 = 10;
    int result;

    //
    result = sum(num1, num2);

    printf("Sum = %d", result);

    return 0;
}

// 
int sum(int a, int b) {
    return a + b;
}
```

### આફ્ટિં:

```
flowchart TD
    A[Main function] --> B[User-defined function]
    B --> A
```

## પ્રશ્ન 5(ક OR) [7 ગુણ]

એરેના ઘટકોનો સરવાળો અને સરેરાશ શોધવા માટેનો C પ્રોગ્રામ લખો.

## જવાબ

```
\#include <stdio.h>

int main() ^{
    int arr[100], n, i;
    int sum = 0;
    float avg;

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

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

    avg = (float)sum / n; //

    printf("Sum of array elements: %d{n}", sum);
    printf("Average of array elements: %.2f", avg);

    return 0;
}
```

## આકૃતિ:

```
flowchart LR
    A([Start]) --> B[/Input size n/]
    B --> C[/Input n elements/]
    C --> D[sum = 0]
    D --> E[i = 0]
    E --> F{i n?}
    F -- Yes --> G["sum += arr[i]"]
    G --> H[i++]
    H --> F
    F -- No --> I[avg = sum / n]
    I --> J[/Output sum, avg/]
    J --> K([End])
```

પગલું	ઓપરેશન	ઉદાહરણ (એરે [5,10,15,20] માટે)
1	ઇનપુટ એરે	[5,10,15,20]
2	sum = 0 ઇનિશિયલાઇઝ	sum = 0
3	દરેક એલિમેન્ટ ઉમેરો	sum = 0+5+10+15+20 = 50
4	સંખ્યા વડે વિભાજીત કરો	avg = 50/4 = 12.5