UIU QUESTION BANK

MID-TERM QUESTION SOLUTIONS

SRUCTURED PROGRAMMING LANGUAGE

CSE 1111

SOLUTION BY

NURUL ALAM ADOR

UPDATED TILL ***SPRING 2024***

nurulalamador.github.io/**UIUQuestionBank**

Index

1. a) **Identify** and **correct** errors in the following code segment.

```
include <stdio>
Int main {
    int Num, a;
    Num = 20%3;
    a = Num+10
    printf("%d %f ", Num, a,);
    return 0;
}
```

Solution:

The code has been rewritten below with the error correction:

```
#include <stdio.h>
int main() {
    int Num, a;
    Num = 20%3;
    a = Num+10;
    printf("%d %d ", Num, a);
    return 0;
}
```

1. b) **Find output** of the following code segment.

```
int a=3, b=4, c=-5, result;
int mod;
result = a * b % c + b;
printf("result = %d\n",result);
if (result >= 0 && result < 10) {
    printf("a =  %d\n", a);
}
else if (result >= 5) {
    printf("b =  %d\n", b);
}
else printf("a = %d\n", a);
```

Solution:

Output:

```
result = 6
a =  3
```

2. a) **Rewrite** the code segment using "**if ... else**" without changing the logical meaning.

```
int num=5, sum=10, i=4, j=9;
```

```
switch(num) {
    case 1: sum *= 3;
    case 2:
    case 3: sum += --j * 2;
            i--; break;
    case 4: sum = ++i * j--;
            break;
    case 5: break;
            i += 10;
    default: sum *= i++ / j--;
             i=i % j; break;
}
```

Solution:

The code has been rewritten below using "if ... else" :

```
int num=5, sum=10, i=4, j=9;

if (num == 1) {
    sum *= 3;
    sum += --j * 2;
    i--;
}
else if (num == 2 || num == 3) {
    sum += --j * 2;
    i--;
}
else if (num == 4) {
    sum = ++i * j--;
}
else if (num == 5) {

}
else {
    sum *= i++ / j--;
    i=i % j;
}
```

2. b) **Manually trace** the following code segment and show **all the changes** of the variables **i,p,** and **x** in **each step**.

```
#include <stdio.h>
int main() {
    int p=1;
    int x = 490;
    for(int i=1;i<=p;){
        printf("%d %d %d\n",i,p,x);
        if (x % 29 == 0){
            printf("Not a great number!");
            break;
        }
        else {
            x -= 13;
            p += x % 10;
            i += 3;
```

```
        }
    }
    return 0;
}
```

Solution:

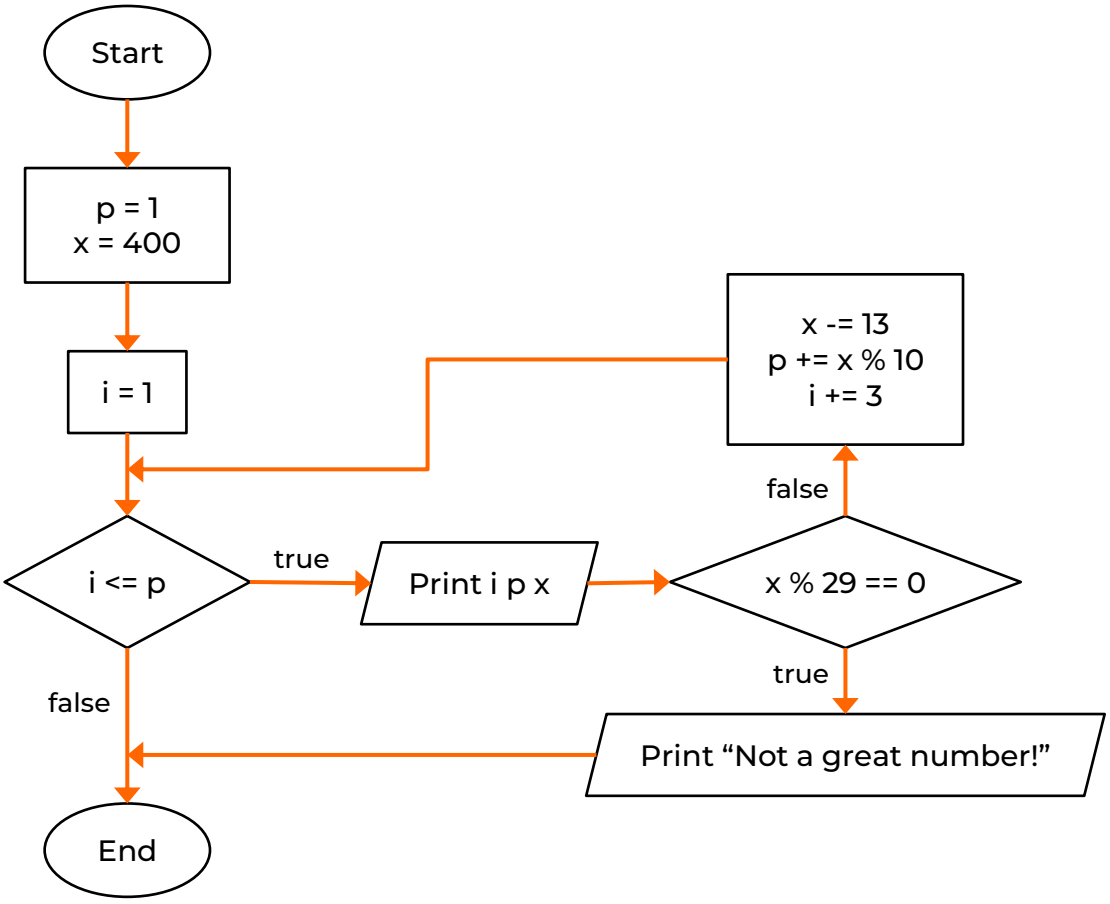
All the changes of the variables **i**, **p**, and **x** in each step has been shown below:

i	p	x	i<=p	x % 29	x -= 13	p += x % 10	i += 3
1	1	490	1<=1 (True)	23 == 0 (False)	477	1+7 = 8	4
4	8	477	4<=8 (True)	13 == 0 (False)	464	8+4 = 12	7
7	12	464	7<=12 (True)	0 == 0 (True)			

2. c) Draw a **flow chart** for the given code segment in **Q.2(b)**.

Solution:

The flow chart has been drawn below:



3. a) Write a **C program** to print the following pattern of **digit '2'**. Take **n** as user input where n is **odd** and **n>=5**.

[P.T.O]

Sample input	n=5
Sample output	***** * ***** * *****

Solution:

The program has been written below:

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            if(i == 0 || i == n-1 || i == n/2) {
                printf("*");
            }
            else if(i < n/2) {
                if(j == n-1) {
                    printf("*");
                }
                else {
                    printf(" ");
                }
            }
            else {
                if(j == 0) {
                    printf("*");
                }
                else {
                    printf(" ");
                }
            }
        }
        printf("\n");
    }
    return 0;
}
```

3. b) Replace the **"outer" while** loop with **"for"** and the **"nested" for** loop with **"while"** loop in the following code without changing the logical meaning of the program.

```
int i=0, count = 0;
int n = 12345;
while (n != 0) {
    printf("%d", n % 10);
    count++;
    for(; i<count; i++) {
        printf("%d", n/= 10);
    }
    printf ("\n");
}
```

Solution:

The code has been rewritten below as per the question:

```
int i=0, count = 0;
int n = 12345;
for (; n != 0; ) {
    printf("%d", n % 10);
    count++;
    while(i<count) {
        printf("%d", n/= 10);
        i++;
    }
    printf("\n");
}
```

4. **Manually trace** the given code segment below. Show the changes of all the variables ***i, hi, hlw*** and array ***arr*** elements in each step.

```
int hi = 0, hlw = 10;
int arr[4] = {10, 20, 30, 40};
for(int i=4; i<=hlw; i++) {
    arr[hi] = arr[hi+1] - 5;
    hlw -= 2;
}
```

Solution:

All the changes of the variables **`i`**, **`hi`**, **`hlw`**, and array **`arr`** elements in each step has been shown below:

hi	hlw	arr	i	i<=hlw	arr[hi] = arr[hi+1]-5	hlw -= 2	i++
0	10	{10, 20, 30, 40}	4	4<=10 (True)	arr[0] = arr[1]-5 = 20-5 = 15	8	5
0	8	{15, 20, 30, 40}	5	4<=8 (True)	arr[0] = arr[1]-5 = 20-5 = 15	6	6
0	6	{15, 20, 30, 40}	6	6<=6 (True)	arr[0] = arr[1]-5 = 20-5 = 15	4	7
0	4	{15, 20, 30, 40}	7	7<=4 (False)			

5. Take an array as **input** of size N. Then take **another number** as **input in K**. Your task is to **add** this **number** to the **even indexed elements**, and **subtract** from the **odd indexed elements**.

Sample Input	Sample Output
N=5 Array Elements: 10 20 30 40 50 K=4	14 16 34 36 54

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    int N, K;

    printf("N=");
    scanf("%d", &N);

    int arr[N];
    printf("Array Elements: ");
    for(int i=0; i<N; i++) {
        scanf("%d", &arr[i]);
    }

    printf("K=");
    scanf("%d", &K);

    for(int i=0; i<N; i++) {
        if(i%2 == 0) {
            arr[i] += K;
        } else {
            arr[i] -= K;
        }
    }

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

    return 0;
}
```

5. Write a program which will take input of **N x N numbers** in a **2D** array A. Now **swap all the elements** in the **first** and **last column** within the array and finally print the array.

Sample Input	Sample Output
3 1 4 7 2 8 3 5 6 0	7 4 1 3 8 2 0 6 5

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    int A[n][n];
    for(int i=0; i<n; i++) {
```

```
        for(int j=0; j<n; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    for (int i=0; i<n; i++) {
        int temp = A[i][0];
        A[i][0] = A[i][n-1];
        A[i][n-1] = temp;
    }

    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

1. a) Which of the following variable names are **invalid** and **why**?

`(i) int_a  (ii) _num  (iii) 99p  (iv) "my_val"`

Solution:

These variable names are invalid from following list:

- `99p`
  Reason: Variable names cannot start with number
- `"my_val"`
  Reason: Variable names cannot have quotation mark

1. b) **Compute** the values of the variables **a, b, c,** and **d**. ASCII codes: A-65, a-97, 0-4.

```
(i)   float a = 101/7;
(ii)  float b = (float)(3%5);
(iii) float c = 23>43 || 6!=6;
(iv)  double result = 12 + (1 * '3');
```

Solution:

The values of the given variables have been written below:

(i) `a = 14.000000`
(ii) `b = 3.000000`
(iii) `c = 0.000000`
(iv) `result = 63.000000`

1. c) **Find outputs** of the following code segment for (i) **num = 2.3**, and (ii) **num = 127**.

```
int num;
scanf("%d", &num);
if (num % 2 != 0)  {
    printf("Mashrafe\n");
}
if (num < 100)  {
    printf("Shakib\n");
}else if (num >= 100){
    printf("Mahmudullah\n");
}
if (num >= 0 && num < 5){
    printf("Imrul\n");
}else if (num >= 0 && num <= 49){
    printf("Tamim\n");
}else{
    printf("Rubel");
}
```

Solution:

[P.T.O]

(i) Output for `num = 2.3` :

```
Shakib
Imrul
```

(ii) Output for `num = 127` :

```
Mashrafe
Mahmudullah
Rubel
```

2. a) **Rewrite** the code segment using "**if ... else**" without changing the logical meaning.

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);
int result=a--/b++;
switch(a+b){
    case 1:
        result+=a/c*2;
        b++;
    case 2:
    case 3:
        result=a*c/b;
        a++;
    case 4: break;
            a=2;
    default: result=5;
}
printf("%d %d %d %d", a,b,c,result);
```

Solution:

The code has been rewritten below using "if ... else" :

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);
int result=a--/b++;
if (a+b == 1) {
    result+=a/c*2;
    b++;
    result=a*c/b;
    a++;
}
else if (a+b == 2 || a+b == 3) {
    result=a*c/b;
    a++;
}
else if (a+b == 4) {

}
else {
    result=5;
}
printf("%d %d %d %d", a,b,c,result);
```

2. b) **Manually trace** the following code segment and show **all** the changes of the variables ***start, end, i, count*** in each step.

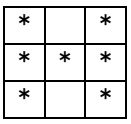
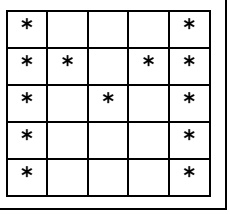
```
int start=105,end=112,count=0;
for(int i=end;  i>=start;  i--){
    if(end%2 != 0){
        count++;
        start++; end+2;
    }else{
        end--; start+1;
    }
}
```

Solution:

All the changes of the variables **`start`**, **`end`**, **`i`** and **`count`** in each step has been shown below:

start	end	count	i	i>=start	end%2 != 0	if	else	i--
105	112	0	112	112 >= 105 (True)	0 != 0 (False)		end--	111
105	111	0	111	111 >= 105 (True)	1 != 0 (True)	count++ start++		110
106	111	1	110	110 >= 106 (True)	1 != 0 (True)	count++ start++		109
107	111	2	109	109 >= 107 (True)	1 != 0 (True)	count++ start++		108
108	111	3	108	108 >= 108 (True)	1 != 0 (True)	count++ start++		107
109	111	4	107	107 >= 109 (False)				

3. a) Write a C program to display the following 'M' pattern for n. For example, for n = 3, & n = 5 the output pattern will be as follows. You must program for n, not for 3 or 5.

Sample input	**n=3**	**n=5**
Sample output		

Solution:

The program has been written below:

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    for(int i=0; i<n; i++) {
```

```
        for(int j=0; j<n; j++) {
            if(j == 0 || j == n-1) {
                printf("* ");
            }
            else if(i <= n/2) {
                if(j == i || j == n-1-i) {
                    printf("* ");
                }
                else {
                    printf("  ");
                }
            }
            else {
                printf("  ");
            }
        }
        printf("\n");
    }
    return 0;
}
```

3. b) Replace the **"outer" for** loop using **"while"** loop and the **"inner" for** loop using **"do while"** loop in the following code without changing the logical meaning of the program.

```
int a=10, b=20, count=0;
for(int i=b;i>=a;i--){
    for(int j=a;j<=b;j++){
        printf("%d ",j);
    }
    if(b%2!=0){
        printf("%d \n",a);
    }else{
        printf("%d \n",b);
    }
}
```

Solution:

The code has been rewritten below as per the question:

```
int a=10, b=20, count=0;
int i=b;
while (i >= a) {
    int j=a;
    do {
        printf("%d ", j);
        j++;
    } while (j <= b);

    if (b%2!=0) {
        printf("%d \n", a);
    } else {
        printf("%d \n", b);
    }
```

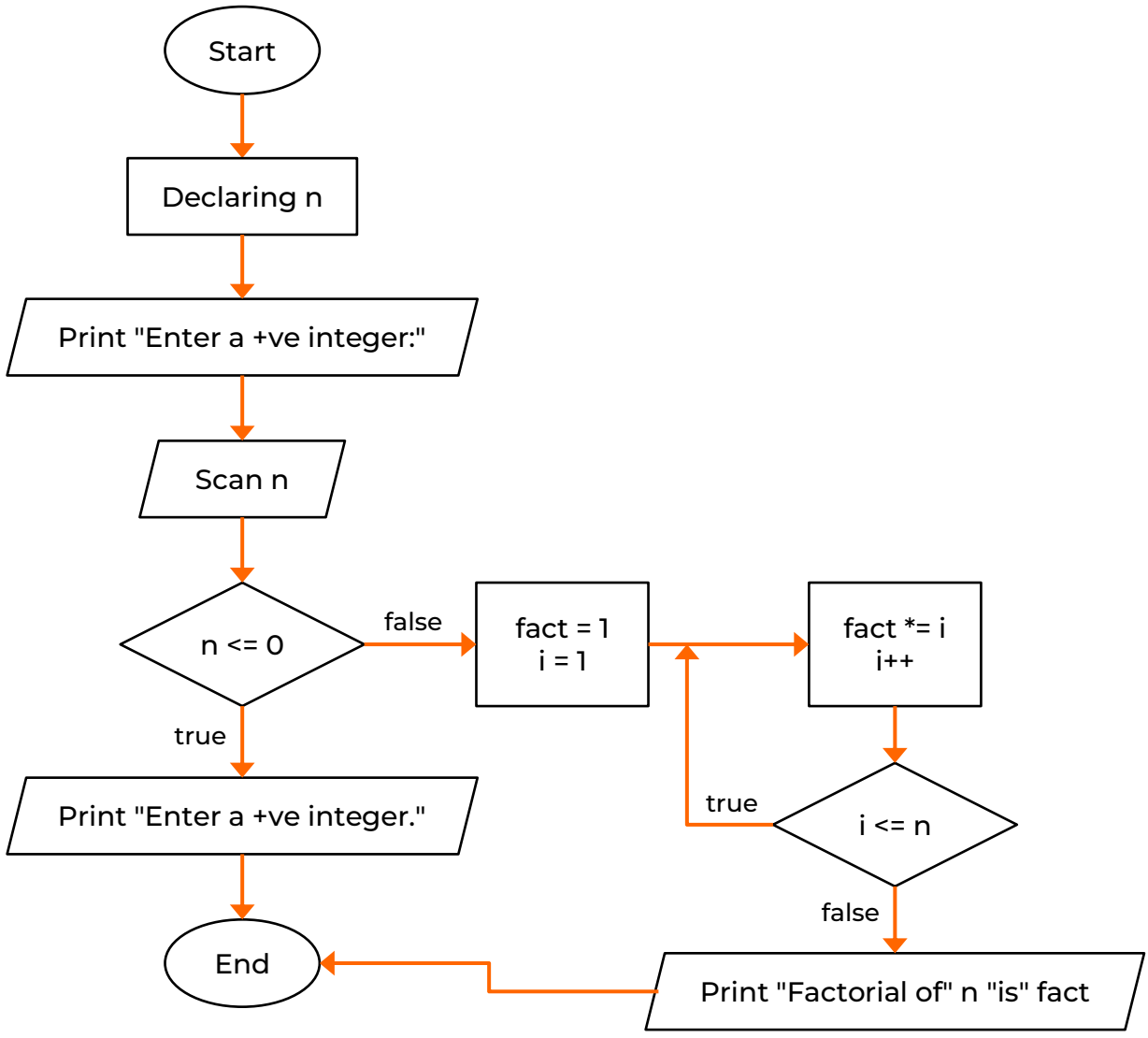
```
    i--;
}
```

4. a) Draw a flow chart for given code segment.

```
int n;
printf("Enter a +ve integer:");
scanf("%d", &n);
if (n <= 0) {
    printf("Enter a +ve integer.");
}else {
    int fact = 1, i = 1;
    do {
        fact *= i;
        i++;
    } while (i <= n);
    printf("Factorial of %d is %d", n, fact);
}
```

Solution:

The flow chart has been drawn below:



4. b) Write a C program to **take** input of all the bank account balance of ***n*** clients of a bank. **Remove** any balance less than 500.00 taka. **Now, display** all the balances.

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    float balances[n];
    for(int i=0; i<n; i++) {
        scanf("%f", &balances[i]);
    }

    for(int i=0; i<n; i++) {
        if (balances[i] < 500.00) {
            for (int j=i; j<n-1; j++) {
                balances[j] = balances[j + 1];
            }
            n--;
            i--;
        }
    }

    for(int i=0; i<n; i++) {
        printf("%f Taka\n", balances[i]);
    }

    return 0;
}
```

5. a) **Manually trace** the given code segment. Show the changes of all the variables ***i,*** and array ***ara*** elements in each step.

```
int ara[5] = { 8,6,2,4,7 };
for(int i = 1; i < 5; i += 2){
    ara[i] =  3 * ara[i - 1];
}
for(int i = 1; i < 5; i++){
    if(i % 2 == 0){
        ara[i] = i * 4 + ara[i-1];
    }
}
```

Solution:

All the changes of the variables **`i`** and elements of array **`ara`** in each step has been shown below:

ara	i	i < 5	ara[i] = 3*ara[i-1]	i+=2	i%2 == 0	ara[i] = i*4+ara[i-1]	i++
{ 8, 6, 2, 4, 7 }	1	1 < 5 (True)	arr[1] = 3*arr[0] = 3*8 = 24	3			

{ 8, 24, 2, 4, 7 }	3	3 < 5 (True)	arr[3] = 3*arr[2] = 3*2 = 6	5			
{ 8, 24, 2, 6, 7 }	5	5 < 5 (False)					
"	1	1 < 5 (True)			1 == 0 (False)		2
"	2	2 < 5 (True)			0 == 0 (True)	arr[2] = 2*4+ara[1] = 8+24 = 32	3
{ 8, 24, 32, 6, 7 }	3	3 < 5 (True)			1 == 0 (False)		4
"	4	4 < 5 (True)			0 == 0 (True)	arr[4] = 4*4+arr[3] = 16+6 = 22	5
{ 8, 24, 32, 6, 22 }	5	5 < 5 (False)					

5. b) **Manually trace** the given code segment and show the changes of all the variables ***row, col,*** and ***sum*** in each step.

```
int row, col, sum = 0;
int A[][3]={{1,2,3}, {11,5,6}, {12,7,9}, {8,13,4}};
for(row=0; row<4; row++){
    for(col=0; col<3; col++){
        if(col>row)  {
            sum += A[row][col];
        }
    }
}
```

Solution:

All the changes of the variables **`row`**, **`col`** and **`sum`** in each step has been shown below:

Here, A =

1	2	3
11	5	6
12	7	9
8	13	4

sum	row	row<4	col	col<3	col>row	sum += A[row][col]	col++	row++
0	0	0<4 (True)	0	0<3 (True)	0>0 (False)		1	
"	"		1	1<3 (True)	1>0 (True)	sum = sum+A[0][1] = 0+2 = 2	2	
2	"		2	2<3 (True)	2>0 (True)	sum = sum+A[0][2] = 2+3 = 5	3	
5	"		3	3<3 (False)				1

"	1	1<4 (True)	0	0<3 (True)	0>1 (False)		1	
"	"		1	1<3 (True)	1>1 (False)		2	
"	"		2	2<3 (True)	2>1 (True)	sum = sum+A[1][2] = 5+6 = 11	3	
11	"		3	3<3 (False)				2
11	2		0	0<3 (True)	0>2 (False)		1	
"	"		1	1<3 (True)	1>2 (False)		2	
"	"		2	2<3 (True)	2>2 (False)		3	
"	"		3	3<3 (False)				3
"	3	3<4 (True)	0	0<3 (True)	0>3 (False)		1	
"	"		1	1<3 (True)	1>3 (False)		2	
"	"		2	2<3 (True)	2>3 (False)		3	
"	"		3	3<3 (False)				4
11	4	4<4 (False)						

Summer 2023

1. a) Which of the following variable names are **invalid** and **why**?

`(i) is-val (ii) a1234 (iii) while (iv) _1num_new (v) CSE 1111`

Solution:

These variable names are invalid from following list:

- `is-val`
  Reason: Variable names cannot have hyphens
- `while`
  Reason: Variable names cannot be a keyword
- `CSE 1111`
  Reason: Variable names cannot have spaces

1. b) **Compute** the values of the variables **a, b, c, d** and **result**.

```
(i)   float a = 22/4;
(ii)  int b = 2%7;
(iii) int e = 4, c = 11 + ++e;
(iv)  int d = 2==3? 7:9;
(v)   double result = 3!=5;
```

Solution:

The values of the given variables have been written below:

(i) `a = 5.000000`
(ii) `b = 2`
(iii) `c = 16`
(iv) `d = 9`
(v) `result = 1.000000`

1. c) **Find outputs** of the following code segment for (i) **a = 0, b = 0**, and (ii) **a = -1, b = -7**:

```
#include<stdio.h>
void main(){
    int a, b;
    scanf("%d%d", &a, &b);
    if(!(a-b) && ++a)
        printf("Pattern\n");
    if((a>0&&b>0)||(a<0&&b<0)){
        printf("Fizz\n");
        if(a>0)
            printf("Positive\n");
        return 0;
        if(b<0)
            printf("Negative\n");
    }
    else if(a>0 && b<0)
        printf("Buzz\n");
```

```
    else printf("FizzBuzz\n");
}
```

Solution:

(i) Output for `a = 0, b = 0`:

```
Pattern
FizzBuzz
```

(ii) Output for `a = -1, b = -7`:

```
Fizz
```

2. a) **Rewrite** the code segment using "**if … else**" without changing the logical meaning.

```
char rank;
scanf("%c", &rank);
int bonus = 0;
switch(rank) {
    case 'p':bonus += 20;
    case 'g':bonus += 20;
    case 's':bonus += 20;
             break;
    default: bonus += 10;
}
printf("\n%d", bonus);
```

Solution:

The code has been rewritten below using "if … else" :

```
char rank;
scanf("%c", &rank);
int bonus = 0;

if(rank == 'p') {
    bonus += 20;
    bonus += 20;
    bonus += 20;
}
else if(rank == 'g') {
    bonus += 20;
    bonus += 20;
}
else if(rank == 's') {
    bonus += 20;
}
else if(rank == 's') {
    bonus += 10;
}
printf("\n%d", bonus);
```

2. b) **Manually trace** the following code segment and show the changes of the variables

i, j, n in each step.

```
#include <stdio.h>
void main() {
    int i = 2, n = 10, j=0;
    for(j = n; j > i; j--) {
        if(j % 2 == 0) i++;
        else n--;
    }
    i += 2;
}
```

Solution:

All the changes of the variables **i**, **j** and **n** in each step has been shown below:

i	n	j	j > i	j % 2 == 0	i++	n--	j--	i +=2
2	10	0						
"	"	10	10 > 1 (True)	0 == 0 (True)	3		9	
3	"	9	9 > 3 (True)	1 == 0 (False)		9	8	
"	9	8	8 > 3 (True)	0 == 0 (True)	4		7	
4	"	7	7 > 4 (True)	1 == 0 (False)		8	6	
"	8	6	6 > 4 (True)	0 == 0 (True)	5		5	
5	"	5	5 > 5 (False)					5+2 = 7
7	8	5						

3. a) **Write a C program** that takes an integer **n** as input from the user and **prints** a specific pattern given as follows. For example, for **n = 4**, the output pattern will be as follows. You must program for n, NOT for 4.

```
*****
*****
*****
*****
```

Solution:

The program has been written below:

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    for(int i=0; i<n; i++) {
```

```
        for(int j=0; j<n-i-1; j++) {
            printf(" ");
        }
        printf("*****\n");
    }
    return 0;
}
```

3. b) **Replace** all the **"for" loops** in the following code using only **"while" loops** without changing the logical meaning of the program.

```
int arr[10]= {0};
int k = 15,
for(int i=1; i<6; i+=2){
    arr[i] = ++k-2;
    k++;
}
int c = 0;
```

```
for(int i=6; i<10; i++)
    for(int j=9; j>=i; j--){
        arr[j] = ++c;
    }
for(int i=0; i<10; i++){
    if(i%2==0)  arr[i] = ++k;
}
```

Solution:

The code has been rewritten below as per the question:

```
int arr[10]= {0};
int k = 15,
int i = 1;
while(i<6){
    arr[i] = ++k-2;
    k++;
    i+=2;
}
int c = 0;
```

```
int i = 6;
while(i<10){
    int j = 9;
    while(j>=i){
        arr[j] = ++c;
        j--;
    }
    i++;
}
i = 0;
while(i<10){
    if(i%2==0)  arr[i] = ++k;
    i++;
}
```

4. a) **Write a C program** that takes ***n*** number of **integers as input** into an array of size ***N***, where ***n*** is an odd number and ***n<=N***. Your task is to **reverse the first half** array elements & the **last half** array elements, keeping **only the middle element** intact.

Intial Array Elements	Final Array Elements
1 2 3 4 5 6 7	3 2 1 **4** 7 6 5
10 20 30 40 50	20 10 **30** 50 40
9 8 7	9 **8** 7

Solution:

The program has been written below:

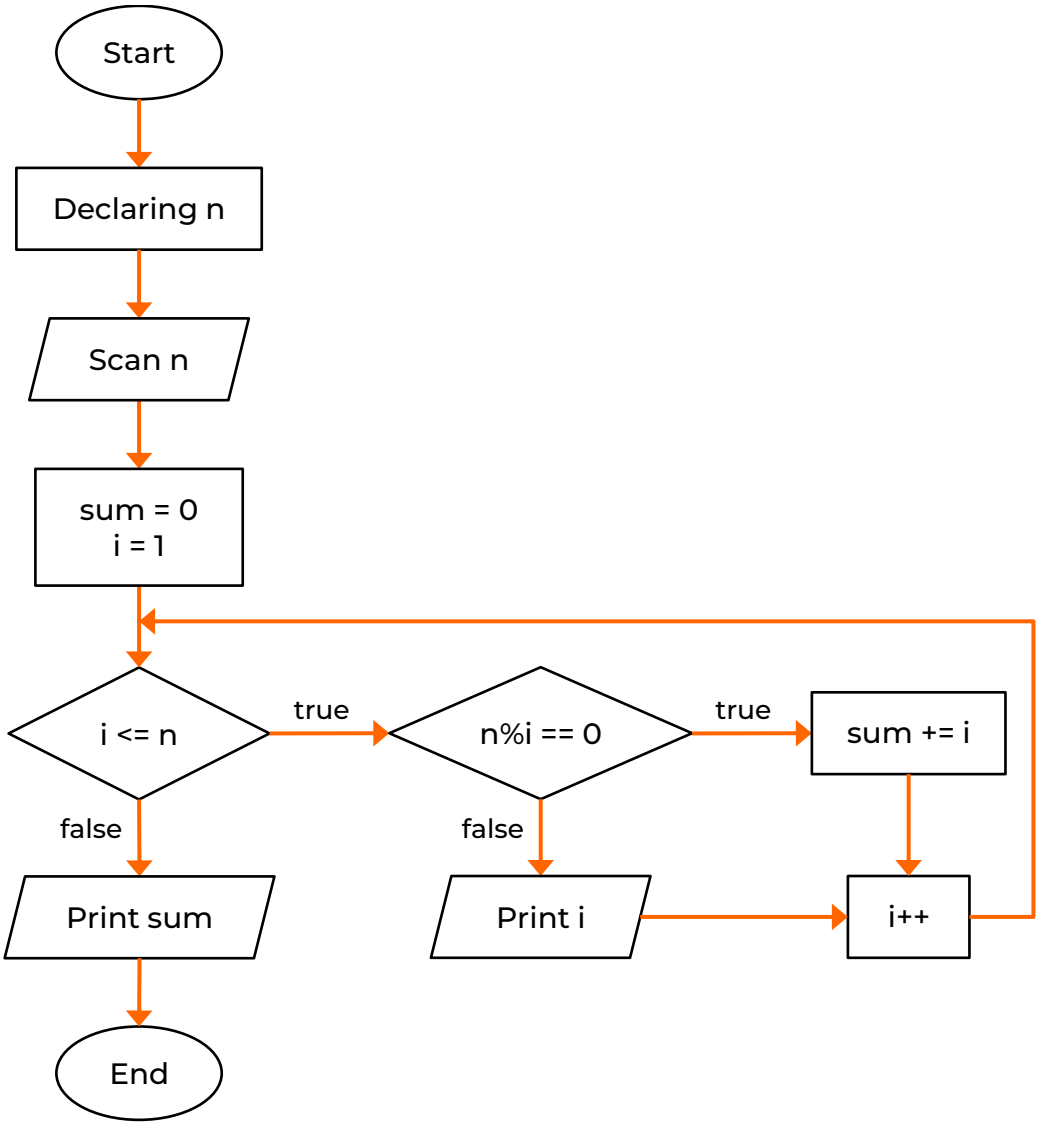
```c
#include <stdio.h>

int main() {
    int arr[9999]; //Let N=9999

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

    int start, end;

    start = 0;
    end = (n/2)-1;
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }

    start = (n/2)+1;
    end = n-1;
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }

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

    return 0;
}
```

4. b) **Draw a flow chart** to take an integer as input. Then, display its **odd factors** and calculate the **sum** of its **even factors**. Hint: any integer number is a multiple of any of its factors.

Sample input	Sample output
20	1 5 [Odd factors] 36 [Sum of even factors: 2+4+10+20=36]
28	1 7 [Odd factors] 48 [Sum of even factors: 2+4+14+28=48]

Solution:

The flow chart has been drawn below:



5. a) **Manually trace** the given code segment. Show the changes of all the variables **i, j, jump** and array **array A and B elements** in each step.

```
int A[4]={3, 2, 1};
int B[4]={10, 20, 30};
int jump=100;
for(int i=0; i<3; i++){
    jump = A[i] * 2;
    for(int j=0; j<3; j++){
        B[i] = A[i] + B[i];
        jump = B[i]/2;
    }
    A[i]++;
}
```

Solution:

All the changes of the variables `i`, `j`, `jump` and elements of array `A`, `B` in each step has been shown below:

A	B	jump	i	i<3	jump = A[i]*2	j	j<3	B[i] = A[i]+B[i]	jump = B[i]/2	A[i]++

3 2 1	10 20 30	100	0	0<3 (True)	jump = A[0]*2 = 3*2 = 6	0	0<3 (True)	B[0] = A[0]+B[0] = 3+10 = 13	jump = B[0]/2 = 13/2 = 6	
3 2 1	13 20 30	6	"			1	1<3 (True)	B[0] = A[0]+B[0] = 3+13 = 16	jump = B[0]/2 = 16/2 = 8	
3 2 1	16 20 30	8	"			2	2<3 (True)	B[0] = A[0]+B[0] = 3+16 = 19	jump = B[0]/2 = 19/2 = 9	
3 2 1	19 20 30	9	"			3	3<3 (False)			A[0] = 4
4 2 1	22 20 30	"	1	1<3 (True)	jump = A[1]*2 = 2*2 = 4	0	0<3 (True)	B[1] = A[1]+B[1] = 2+20 = 22	jump = B[1]/2 = 22/2 = 11	
4 2 1	22 22 30	11	"			1	1<3 (True)	B[1] = A[1]+B[1] = 2+22 = 24	jump = B[1]/2 = 25/2 = 12	
4 2 1	22 24 30	12	"			2	2<3 (True)	B[1] = A[1]+B[1] = 2+24 = 26	jump = B[1]/2 = 26/2 = 13	
4 2 1	22 26 30	13	"			3	3<3 (False)			A[1] = 3
4 3 1	22 26 30	"	2	2<3 (True)	jump = A[2]*2 = 1*2 = 2	0	0<3 (True)	B[2] = A[2]+B[2] = 1+30 = 31	jump = B[1]/2 = 31/2 = 15	
4 3 1	22 26 31	15	"			1	0<3 (True)	B[2] = A[2]+B[2] = 1+31 = 32	jump = B[1]/2 = 32/2 = 16	
4 3 1	22 26 32	16	"			2	0<3 (True)	B[2] = A[2]+B[2] = 1+32 = 33	jump = B[1]/2 = 33/2 = 16	
4 3 1	22 26 33	16	"			3	3<3 (False)			A[1] = 2
4 3 2	22 26 33	16	3	3<3 (False)						

5. b) **Manually trace** the given code segment and show the changes of all the variables **i, j, x** and **sum** in each step.

```
int arr[][4]={{5, 7, 3, 13},
              {31, 2, 11, 23},
              {17, 19, 43, 53},
              {37, 47, 29, 61}};
```

```
int n=4, sum=0, x = 0;
for(int i = 0; i<n; i++){
    for(int j = 0; j<n; j++){
        if(j==n-1 || i+j==n-1){
            x = arr[i][j];
            sum+=x;
        }
    }
}
```

Solution:

All the changes of the variables **i**, **j**, **x** and **sum** in each step has been shown below:

Here, n = 4

arr =

5	7	3	13
31	2	11	23
17	19	43	53
37	47	29	61

sum	x	i	i<n	j	j<n	j==n-1 || i+j==n-1	x = arr[i][j]	sum+=x	j++	i++
0	0	0	0<4 (True)	0	0<4 (True)	0 == 3 || 0 == 3 (False)			1	
"	"	"		1	1<4 (True)	1 == 3 || 1 == 3 (False)			2	
"	"	"		2	2<4 (True)	2 == 3 || 2 == 3 (False)			3	
"	"	"		3	3<4 (True)	3 == 3 || 3 == 3 (True)	x = arr[0][3] = 13	sum = 0+13 = 13	4	
13	13	"		4	4<4 (False)					1
"	"	1	1<4 (True)	0	0<4 (True)	0 == 3 || 1 == 3 (False)			1	
"	"	"		1	1<4 (True)	1 == 3 || 2 == 3 (False)			2	
"	"	"		2	2<4 (True)	2 == 3 || 3 == 3 (True)	x = arr[1][2] = 11	sum = 13+11 = 24	3	
24	11	"		3	3<4 (True)	3 == 3 || 4 == 3 (True)	x = arr[1][3] = 23	sum = 24+23 = 47	4	
47	23	"		4	4<4 (False)					2
"	"	2	2<4 (True)	0	0<4 (True)	0 == 3 || 2 == 3 (False)			1	

"	"	"		1	1<4 (True)	1 == 3 || 3 == 3 (True)	x = arr[2][1] = 19	sum = 47+19 = 66	2	
66	19	"		2	2<4 (True)	2 == 3 || 4 == 3 (False)			3	
"	"	"		3	3<4 (True)	3 == 3 || 5 == 3 (True)	x = arr[2][3] = 53	sum = 66+53 = 119	4	
119	53	"		4	4<4 (False)					3
"	"	3	3<4 (True)	0	0<4 (True)	0 == 3 || 3 == 3 (True)	x = arr[3][0] = 37	sum = 119+37 = 156	1	
156	37	"		1	1<4 (True)	1 == 3 || 4 == 3 (False)			2	
"	"	"		2	2<4 (True)	2 == 3 || 5 == 3 (False)			3	
"	"	"		3	3<4 (True)	3 == 3 || 6 == 3 (True)	x = arr[3][3] = 61	sum = 156+61 = 217	4	
217	61	"		4	4<4 (False)					4
"	"	4	4<4 (False)	"						

1. a) Which of the following are **invalid** variable names?

(i) `1UIU` (ii) `SPL_2023` (iii) `char` (iv) `SPL$` (v) `My-Course`

Solution:

These variable names are invalid from following list:

- `1UIU`
  Reason: Variable names cannot start with a number
- `char`
  Reason: Variable names cannot be a keyword
- `My-Course`
  Reason: Variable names cannot have hyphens

1. b) **Compute** the values of the variables **a, b, c** and **d**.

```
int a = (float)15/4;
float b = a++*a--;
int c = (a>b || a==1+2)*2;
float d = a/c;
```

Solution:

The values of the given variables have been written below:

```
a = 3
b = 12.000000
c = 2
d = 1.000000
```

1. c) **Find the outputs** of the following program for (i) **b=10**, and (ii) **b=2**:

```
#include <stdio.h>
int main() {
    int b;
    scanf("%d", &b);
    if(b >= 10) {
        printf("SPL\n");
        b--;
    }
    if(b < 10) {
        printf("Spring\n");
        b--;
    }
    else if((b>=3) || (b<10))
        printf("2023\n");
    else if(b>=3 && b<10)
        printf("Happy Coding!");
    else
        printf("Huh!");
    return 0;
}
```

Solution:

(i) Output for `b = 10` :

```
SPL
Spring
```

(ii) Output for `b = 2` :

```
Spring
```

2. a) **Rewrite** the following code segment using **"switch … case"** without changing the logical meaning.

```
int n, a;
scanf("%d %d", &n, &a);
if(n>a) {
    if(n-a>5) {
        printf("Difference is greater than 5 \n");
    }
    else {
        printf("Difference is less than or equal to 5 \n");
    }
}
else {
    printf("Please give a larger value of n \n");
}
```

Solution:

The code has been rewritten below using "switch … case" :

```
int n, a;
scanf("%d %d", &n, &a);

switch (n>a) {
    case 1:
        switch (n-a>5) {
            case 1:
                printf("Difference is greater than 5 \n");
                break;
            default:
                printf("Difference is less than or equal to 5 \n");
                break;
        }
        break;
    default:
        printf("Please give a larger value of n \n");
        break;
}
```

2. b) **Manually trace** the following code segment and show the changes of the values of variables **i, j, result, x, y** in each step.

```
int result = 5, i, x = 2, y = 2;
for(int j = 8; j > 3; --j) {
    i = (j * result) / x;
    result += y;
    x += (y-2);
    y++;
}
```

Solution:

All the changes of the variables **i**, **j**, **result**, **x**, **y** in each step has been shown below:

result	i	x	y	j	j > 3	i = (j*result)/x	result += y	x += (y-2)	y++	--j
5		2	2	8	8 > 3 (True)	i = (8*5)/2 = 20	result = 5+2 = 7	x = 2+(2-2) = 2	3	7
7	20	2	3	7	7 > 3 (True)	i = (7*7)/2 = 24	result = 7+3 = 10	x = 2+(3-2) = 3	4	6
10	24	3	4	6	6 > 3 (True)	i = (6*10)/3 = 20	result = 10+4 = 14	x = 3+(4-2) = 5	5	5
14	20	5	5	5	5 > 3 (True)	i = (5*14)/5 = 14	result = 14+5 = 19	x = 5+(5-2) = 8	6	4
19	14	8	6	4	4 > 3 (True)	i = (4*19)/8 = 9	result = 19+6 = 25	x = 8+(6-2) = 12	7	3
25	9	12	7	3	3 > 3 (False)					

3. a) **Replace** the **nested "for" loops** in the following code using only **nested "do ... while" loops** without changing the logical meaning of the program:

```
int main() {
    int weeks = 2, days_in_week = 7;
    for (int i = 1; i <= weeks; ++i) {
        printf("Week: %d\n", i);
        for (int j = 1; j <= days_in_week; ++j) {
            if (i%2 == 0) {
                if(j%2 == 0)
                    printf("    Day: %d\n", j);
            }
            else{
                if(j%2 != 0)
                    printf("    Day: %d\n", j);
            }
        }
    }
    return 0;
}
```

Solution:

The code has been rewritten below as per the question:

```c
int main() {
    int weeks = 2, days_in_week = 7;
    int i = 1;
    do {
        printf("Week: %d\n", i);
        int j = 1;
        do {
            if (i%2 == 0) {
                if (j%2 == 0)
                    printf("    Day: %d\n", j);
            }
            else {
                if (j%2 != 0)
                    printf("    Day: %d\n", j);
            }
            j++;
        } while (j <= days_in_week);

        i++;
    } while (i <= weeks);

    return 0;
}
```

3. b) **Write a C program** that takes an integer **n** as input from the user and **prints** the following pattern using nested loop.

Sample input, n	Sample output

3	`  2`
	`  4  6  4`
	`6  8 10  8  6`

5	`  2`
	`  4  6  4`
	`  6  8 10  8  6`
	`  8 10 12 14 12 10  8`
	`10 12 14 16 18 16 14 12 10`

Solution:

The program has been written below:

```c
#include<stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    for (int i=1; i<=n; i++) {
        int count = 2*i;
        for (int j=0; j<n-i; j++) {
            printf("   ");
        }
        for (int j=0; j<i; j++) {
            printf("%2d ", count);
            count += 2;
        }
```

```
        count -= 2;
        for (int j=0; j<i-1; j++) {
            count -= 2;
            printf("%2d ", count);
        }
        printf("\n", i);
    }
    return 0;
}
```

4. a) **Write a C program** to perform the following operations-

i) **Take input** of CGPA of 100 students
ii) **Calculate the average** of the CGPA of the students who achieved more than 3.00
iii) Find out the **highest** and **lowest CGPA** and **how many students** achieved that highest CGPA.

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    float cgpa[100];

    for (int i=0; i<100; i++) {
        scanf("%f", &cgpa[i]);
    }

    int count = 0;
    float sum = 0;
    float highestCount = 0;
    int highest = 0;
    int lowest = 4;

    for (int i=0; i<100; i++) {
        if (cgpa[i] > 3.00) {
            sum += cgpa[i];
            count++;
        }

        if (cgpa[i] > highest) {
            highest = cgpa[i];
            highestCount = 1;
        }
        else if (cgpa[i] == highest) {
            highestCount++;
        }

        if (cgpa[i] < lowest) {
            lowest = cgpa[i];
        }
    }
```

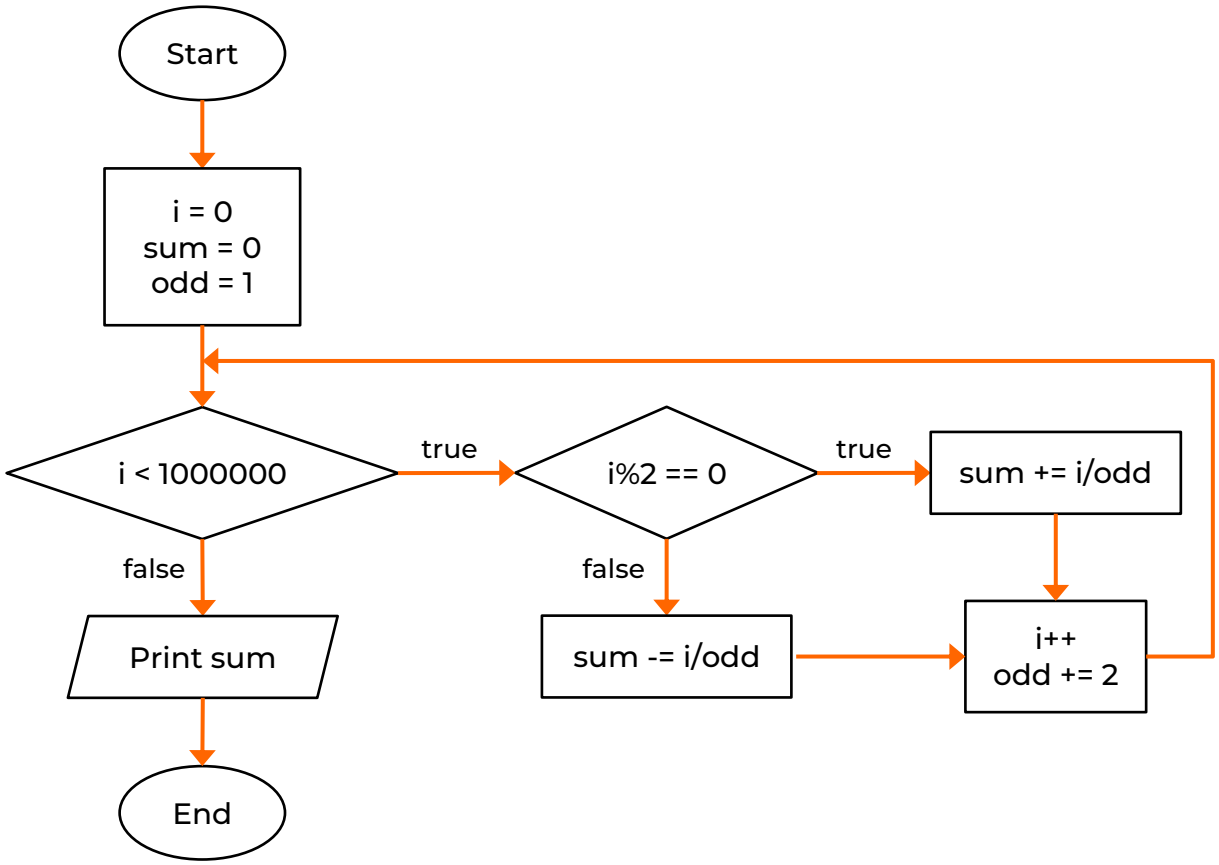
```
    float average = sum/count;
    if (count != 0) {
        printf("Average CGPA of student who achieved more than 3.00:
                                                        %f\n", average);
    } else {
        printf("No students achieved CGPA more than 3.00\n");
    }

    printf("Highest CGPA: %f\n", highest);
    printf("Number of students achieved the highest CGPA: %f\n",
                                                          highestCount);
    printf("Lowest CGPA: %f", lowest);

    return 0;
}
```

4. b) **Draw a flowchart** to calculate **summation of the following series** up to **1000000-th** term.

$$1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}-\cdots$$

Solution:

The flow chart has been drawn below:



5. a) **Manually trace** the given code segment for the following array **"arr"**. Show the changes of all the variables in each step.

```
#include<stdio.h>
int main() {
    int arr[10]= {0};
    int k = 15;
    for(int i=1; i<6; i+=2)
    {
        arr[i] = ++k-2;
        k++;
    }
    int c = 0;
    for(int i=6; i<10; i++)
    {
        for(int j=10; j>=i; j--)
        {
            arr[j] = ++c;
        }
    }
    for(int i=0; i<10; i++)
    {
        if(i%2==0)
        {
            arr[i] = ++k;
        }
    }
}
```

Solution:

All the changes of the variables and elements of array **arr** in each step has been shown below:

Manual tracing for 1st for loop:

arr	k	i	i<6	arr[i] = ++k-2	k++	i+=2
{ 0, NIL, NIL, NIL, NIL, NIL, NIL, NIL, NIL, NIL }	15	1	1<6 (True)	arr[1] = ++15-2 = 16-2 = 14	17	3
{ 0, 14, NIL, NIL, NIL, NIL, NIL, NIL, NIL, NIL }	17	3	3<6 (True)	arr[3] = ++17-2 = 18-2 = 16	19	5
{ 0, 14, NIL, 16, NIL, NIL, NIL, NIL, NIL, NIL }	19	5	5<6 (True)	arr[5] = ++19-2 = 20-2 = 18	21	7
{ 0, 14, NIL, 16, NIL, 18, NIL, NIL, NIL, NIL }	21	7	7<6 (False)			

Manual tracing for 2nd for loop:

arr	c	i	i<10	j	j>=i	arr[j] = ++c	j--	i++
{ 0, 14, NIL, 16, NIL, 18, NIL, NIL, NIL, NIL }	0	6	6<10 (True)	10	10>=6 (True)	arr[10] = ++0 = 1	9	
{ 0, 14, NIL, 16, NIL, 18, NIL, NIL, NIL, NIL }	1			9	9>=6 (True)	arr[9] = ++1 = 2	8	
{ 0, 14, NIL, 16, NIL, 18, NIL, NIL, NIL, 2 }	2			8	8>=6 (True)	arr[8] = ++2 = 3	7	

{ 0, 14, NIL, 16, NIL, 18, NIL, NIL, 3, 2 }	3			7	7>=6 (True)	arr[7] = ++3 = 4	6	
{ 0, 14, NIL, 16, NIL, 18, NIL, 4, 3, 2 }	4			6	6>=6 (True)	arr[6] = ++4 = 5	5	
{ 0, 14, NIL, 16, NIL, 18, 5, 4, 3, 2 }	5			5	5>=6 False			7
"	"	7	7<10 (True)	10	10>=7 (True)	arr[10] = ++5 = 6	9	
{ 0, 14, NIL, 16, NIL, 18, 5, 4, 3, 2 }	6			9	9>=7 (True)	arr[9] = ++6 = 7	8	
{ 0, 14, NIL, 16, NIL, 18, 5, 4, 3, 7 }	7			8	8>=7 (True)	arr[8] = ++7 = 8	7	
{ 0, 14, NIL, 16, NIL, 18, 5, 4, 8, 7 }	8			7	7>=7 (True)	arr[7] = ++8 = 9	6	
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 8, 7 }	9			6	6>=7 (False)			8
"	"	8	8<10 (True)	10	10>=8 (True)	arr[10] = ++9 = 10	9	
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 8, 7 }	10			9	9>=8 (True)	arr[9] = ++10 = 11	8	
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 8, 11 }	11			8	8>=8 (True)	arr[8] = ++11 = 12	7	
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 12, 11 }	12			7	7>=8 (False)			9
"	"	9	9<10 (True)	10	10>=9 (True)	arr[10] = ++12 = 13	9	
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 12, 11 }	13			9	9>=9 (True)	arr[9] = ++13 = 14	8	
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 12, 14 }	14			8	8>=9 (False)			10
"	"	10	10<10 (False)					

Manual tracing for 3rd for loop:

arr	k	i	i<10	i%2==0	arr[i] = ++k-2	i++
{ 0, 14, NIL, 16, NIL, 18, 5, 9, 12, 14 }	21	0	0<10 (True)	0==0 (True)	arr[0] = ++21 = 22	1
{ 22, 14, NIL, 16, NIL, 18, 5, 9, 12, 14 }	22	1	1<10 (True)	1==0 (False)		2
"	"	2	2<10 (True)	0==0 (True)	arr[2] = ++22 = 23	3

{ 22, 14, 23, 16, NIL, 18, 5, 9, 12, 14 }	23	3	3<10 (True)	1==0 (False)		4
"	"	4	4<10 (True)	0==0 (True)	arr[4] = ++23 = 24	5
{ 22, 14, 23, 16, 24, 18, 5, 9, 12, 14 }	24	5	5<10 (True)	1==0 (False)		6
"	"	6	6<10 (True)	0==0 (True)	arr[6] = ++24 = 25	7
{ 22, 14, 23, 16, 24, 18, 25, 9, 12, 14 }	25	7	7<10 (True)	1==0 (False)		8
"	"	8	8<10 (True)	0==0 (True)	arr[8] = ++25 = 26	9
{ 22, 14, 23, 16, 24, 18, 25, 9, 26, 14 }	26	9	9<10 (True)	1==0 (False)		10
"	"	10	10<10 (False)			

5. b) **Manually trace** the following code snippet and **find the final content** of the **2D array "arr"** after the execution of the code.

```
int arr[100][100], i, j, t1 = 0, t2 = 1, t3, x, y, z;
for(i=0; i<5; i++)
{
    x = t1, y = t2, z = t1+t2;
    for(j=0; j<5; j++)
    {
        t3 = t1 + t2;
        arr[j][i] = t3;
        t1 = t2;
        t2 = t3;
    }
    t1 = y;
    t2 = z;
}
```

Solution:

All the changes of the variables in each step has been shown below:

t1	**t2**	**t3**	**x**	**y**	**z**	**i**	**i<5**	**x = t1 y = t2 z = t1+t2**	**j**	**j<5**	**t3 = t1 + t2 arr[j][i] = t3 t1 = t2 t2 = t3**	**t1 = y t2 = z**
0	1					0	0<5 (True)	x = 0 y = 1 z = 1	0	0<5 (True)	t3 = 0+1 = 1 arr[0][0] = 1 t1 = 1 t2 = 1	
1	1	1	0	1	1				1	1<5	t3 = 1+1 = 2	

										(True)	arr[0][1] = 2 t1 = 1 t2 = 2	
1	2	2							2	2<5 (True	t3 = 1+2 = 3 arr[0][2] = 3 t1 = 2 t2 = 3	
2	3	3							3	3<5 (True	t3 = 5 arr[0][3] = 5 t1 = 3 t2 = 5	
3	5	5							4	4<5 (True)	t3 = 8 arr[0][4] = 8 t1 = 5 t2 = 8	
5	8	8							5	5<5 (False)		t1 = 1 t2 = 1
1	1	8				1	1<5 (True)	x = 1 y = 1 z = 2	0	0<5 (True)	t3 = 2 arr[1][0] = 2 t1 = 1 t2 = 2	
1	2	2	1	1	2				1	1<5 (True)	t3 = 3 arr[1][1] = 3 t1 = 2 t2 = 3	
2	3	3							2	2<5 (True	t3 = 5 arr[1][2] = 5 t1 = 3 t2 = 5	
3	5	5							3	3<5 (True	t3 = 8 arr[1][3] = 8 t1 = 5 t2 = 8	
5	8	8							4	4<5 (True)	t3 = 13 arr[1][4] = 13 t1 = 8 t2 = 13	
8	13	13							5	5<5 (False)		t1 = 1 t2 = 2
1	2	13				2	2<5 (True)	x = 1 y = 2 z = 3	0	0<5 (True)	t3 = 3 arr[2][0] = 3 t1 = 2 t2 = 3	
2	3	3							1	1<5 (True)	t3 = 5 arr[2][1] = 5 t1 = 3 t2 = 5	

3	5	5							2	2<5 (True	t3 = 8 arr[2][2] = 8 t1 = 5 t2 = 8	
5	8	8							3	3<5 (True	t3 = 13 arr[2][3] = 13 t1 = 8 t2 = 13	
8	13	13							4	4<5 (True)	t3 = 21 arr[2][4] = 21 t1 = 13 t2 = 21	
13	21	21							5	5<5 (False)		t1 = 2 t2 = 3

By observing the manual tracing so far, we understand that each row of the array is a Fibonacci series of five numbers and shifting one value to the left after each row. Therefore, we can assume the final content of the 2D array without further manual trace. The final content of 2D array **`arr`** is:

arr =

1	2	3	5	8
2	3	5	8	13
3	5	8	13	21
5	8	13	21	34
8	13	21	34	55

1. a) **Rewrite** the following code after **correcting the errors**.

```
#includes <studio.h>
int Main() {
    int a, b, float sum;
    Scanf(“%i”, &a);
    a , b=10;
    a+b =sum;
    Printf(“%d”, &sum);
}
```

Solution:

The code has been rewritten below with the error correction:

```
#include <stdio.h>
int main() {
    int a, b;
    float sum;
    scanf("%i", &a);
    a=b=10;
    sum = a+b;
    printf("%f", sum);
}
```

1. b) **Identify** the **invalid variable names** from the following. **Mention the reasons** that make them invalid.

```
sum_of_digit , switch , calculate sum , ( )value , const , Sum,
                    calculate-sum , 1st_sum
```

Solution:

The following variable names are invalid from following list:

- `switch`
  Reason: Variable names cannot be any keyword
- `calculate sum`
  Reason: Variable names cannot have spaces
- `( )value`
  Reason: Variable names cannot have parentheses
- `const`
  Reason: Variable names cannot be any keyword
- `calculate-sum`
  Reason: Variable names cannot have hyphens
- `1st_sum`
  Reason: Variable names cannot start with number

1. c) **Compute** the values of the variables **a, b, c,** and **d**.

```
int a = 17%7*5;
```

```
float b = (int)(17.0/5);
float c= (int)17/5.0;
int d = (a>b) && c;
```

Solution:

The values of the given variables have been written below:

- a = 15
- b = 3.000000
- c = 3.400000
- d = 1

2. a) **Manually trace** the following C code segment for **num=3** and show the **changes of values of all the variables** in each step.

```
#include <stdio.h>
int main() {
    int num;
    int sum = 10, i =7, j = 2;
    scanf("%d", &num);
    switch(num) {
        case 1:
        case 2:
            sum += --j*2;
            i--;
        case 3:
            sum = ++i*j--;
            break;
        case 4:
            sum *= i++/j--;
            i=i%j;
        default: break;
    }
    printf("%d %d %d",sum,i,j);
    return 0;
}
```

Solution:

Here,

`num = 3`
`sum = 10`
`i = 7`
`j = 2`

For the `switch(num)`, where `num = 3`:

`1 != num` ∴ `case 1` is `false`
`2 != num` ∴ `case 2` is `false`
`3 == num` ∴ `case 3` is `true`

Since `case 3` is `true`, it will execute statements from `case 3`.

Now,

`sum = ++i*j-- = ++7*2-- = 8*2 = 16`
∴ `sum = 16`
∴ `i = 8`
∴ `j = 1`

Since there are **break** after this statement, operations of **switch** will stop here.

Now **printf("%d %d %d",sum,i,j)** will print:

```
16 8 1
```

2. b) **Re-write** the given C code segment in Q.2(a) using the **"if-else"** statement without changing the logical meaning and output.

Solution:

The code has been rewritten below using "if ... else" :

```
#include <stdio.h>
int main() {
    int num;
    int sum = 10, i =7, j = 2;
    scanf("%d", &num);
    if(num == 1 || num == 2) {
        sum += --j*2;
        i--;
        sum = ++i*j--;
    }
    else if(num == 3) {
        sum = ++i*j--;
    }
    else if(num == 4) {
        sum *= i++/j--;
        i=i%j;
    }
    printf("%d %d %d",sum,i,j);
    return 0;
}
```

3. a) Write a complete program to print the following series up to **'Nth'** term and also find the **sum of the series**.

Sample Input	N = 6
Sample Output	0, 5, 18, 39, 68, 105 Sum - 235

Solution:

The program has been written below:

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);

    int a=0;
    int b=0;
    int sum=0;
    for(int i=0; i<n; i++){
```

```
        int term = a*a+b;
        printf("%d, ", term);
        sum += term;
        a += 2;
        b++;
    }
    printf("\nSum = %d", sum);
    return 0;
}
```

3. b) **Manually trace** the following code for '**rows = 3'. Show** changes of **all the variables** in each step.

```
int i, space, rows, k = 0, count = 0, count1 = 0;
for (i = 1; i <= rows; ++i) {
    for (space = 1; space <= rows - i; ++space) {
        printf("  ");
        ++count;
    }
    while (k != 2 * i - 1) {
        if (count <= rows - 1) {
            printf("%d ", i + k);
            ++count;
        } else {
            ++count1;
            printf("%d ", (i + k - 2 * count1));
        }
        ++k;
    }
    count1 = count = k = 0;
    printf("\n");
}
```

Solution:

All the changes of the variables in each step for **`rows = 3`** has been shown below:

count	count1	i	i <= rows	space	space <= rows-i	k	k != 2*i-1	count <= rows-1
0	0	1	1 <= 3 (True)	1	1 <= 2 (True) ++count ++space	0		
1	"	"		2	2 <= 2 (True) ++count ++space	"		
2	"	"		3	3 <= 2 (False)	"		
"	"	"				"	0 != 1 (True)	2 <= 2 (True) ++count ++k

3	"	"				1	1 != 1 (False)	
0	0	"				0		
"	"	2	2 <= 3 (True)	1	1 <= 1 (True) ++count ++space	"		
1	"	"		2	2 <= 1 (False)	"		
"	"	"				"	0 != 3 (True)	1 <= 2 (True) ++count ++k
2	"	"				1	1 != 3 (True)	2 <= 2 (True) ++count ++k
3	"	"				2	2 != 3 (True)	3 <= 2 (False) ++count1 ++k
"	1	"				3	3 != 3 (False)	
0	0	"				0		
"	"	3	3 <= 3 (True)	1	1 <= 0 (False)	"		
"	"					"	0 != 5 (True)	0 <= 2 (True) ++count ++k
1	"					1	1 != 5 (True)	1 <= 2 (True) ++count ++k
2	"					2	2 != 5 (True)	2 <= 2 (True) ++count ++kk
3	"					3	3 != 5 (True)	3 <= 2 (False) ++count1 ++k
"	1	"				4	4 != 5 (True)	3 <= 2 (False) ++count1 ++k

"	2	"				5	5 != 5 (False)	
0	0	"				0		
"	"	4	4 <= 3 (False)			"		

4. a) **Manually trace** the given code segment for the following array assuming **size=5**. **Show** changes of **all the variables** in each step.

```
for(i=0; i<size; i++){
    for(j=i+1; j<size; j++){
        if(arr[i] == arr[j]){
            for(k=j; k<size-1; ++k){
                arr[k] = arr[k+1];
            }
            size--;
            j--;
        }
    }
}
```

A	10	20	10	10	100

Solution:

All the changes of the variables in each step has been shown below:

arr	size	i	i<size	j	j<size	arr[i] == arr[j]	k	k<size-1	size--	j--	j++
{10, 20, 10, 10, 100}	5	0	0<5 (True)	1	1<5 (True)	10 == 20 (False)					
"	"	"		2	2<5 (True)	10 == 10 (True)	2	2<4 (True) arr[2] = 10			
{10, 20, 10, 10, 100}	"	"		"			3	3<4 (True) arr[3] = 100			
{10, 20, 10, 100, 100}	"	"		"			4	4<4 (False)	4	1	2
"	4	"		2	2<4 (True)	10 == 10 (True)	2	2<3 (True) arr[2] = 100			
{10, 20, 100, 100, 100}		"		"			3	3<3 (False)	3	1	2
"	3	"		2	2<3 (True)	10 == 100 (False)					
"	"	"		3	3<3 (False)						
"	"	1	1<3 (True)	2	2<3 (True)	20 == 100 (False)					

"	"			3	3<3 (False)						
"	"	2	2<3 (True)	3	3<3 (False)						
"	"	3	3<3 (False)	"							

4. b) Write a program that **reads a number n**. Now, take **n inputs** in an array named **marks[10]**, here n is less than 10. Now, find the maximum number and its position within the even values of the array.

Sample Input:	**Sample Output:**
6 1 **10** **6** 51 **24** 13	Even value max = 24, found in index 4

Solution:

The program has been written below:

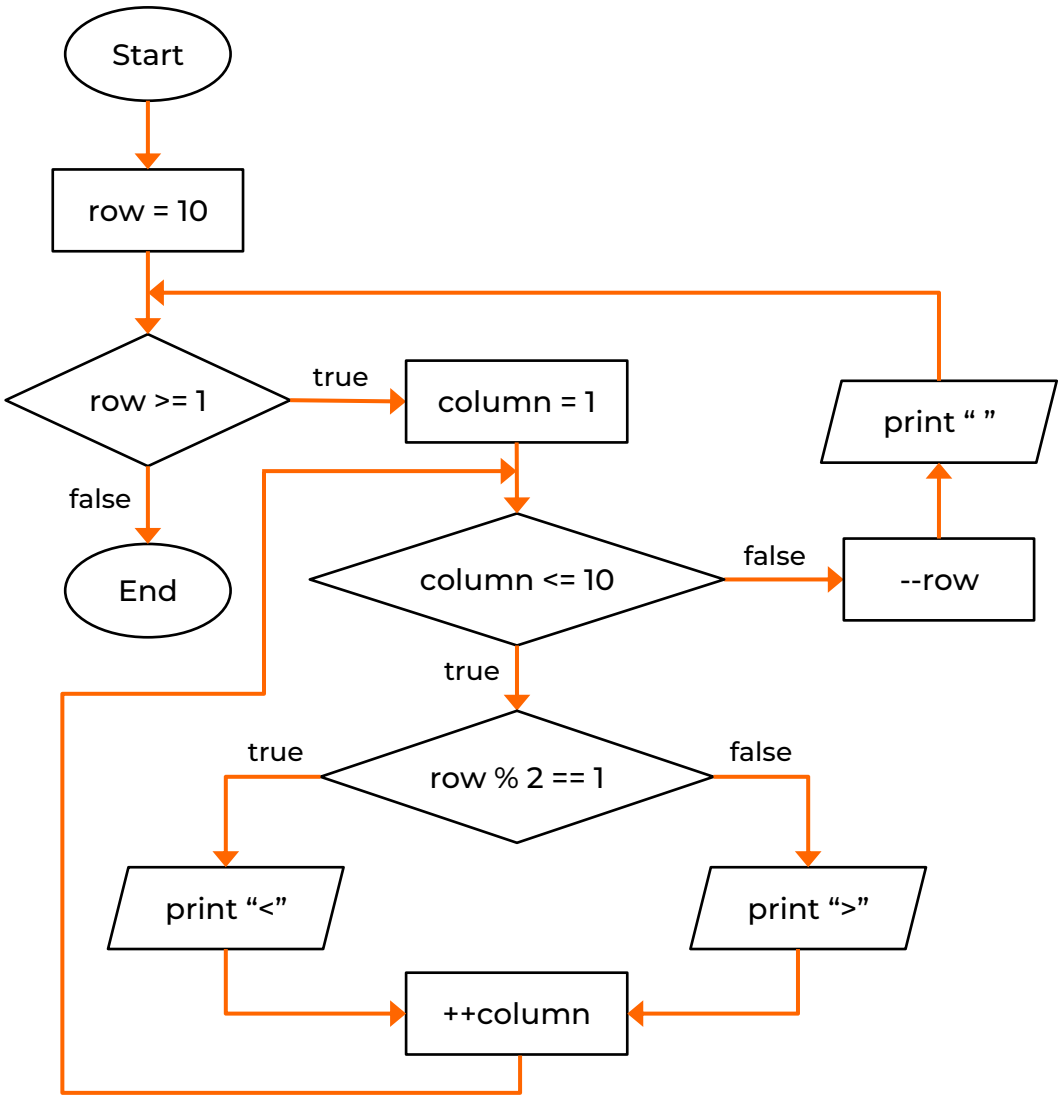
```
#include <stdio.h>

int main() {
    int marks[10];

    int n;
    scanf("%d", &n);

    for (int i=0; i<n; i++) {
        scanf("%d", &marks[i]);
    }

    int max = -1;
    int index = -1;

    for (int i=0; i<n; i++) {
        if (marks[i]%2 == 0 && marks[i] > max) {
            max = marks[i];
            index = i;
        }
    }

    if (index != -1) {
        printf("Even value max = %d, found in index %d", max,
index);
    } else {
        printf("No even number found!");
    }

    return 0;
}
```

5. a) **Draw a flowchart** for the C program given as follows:

```c
#include <stdio.h>
int main(void) {
    int row = 10;
    while (row >= 1) {
        int column = 1;
        while (column <= 10) {
            printf("%s", (row % 2) ? "<": ">");
            ++column;
        }
        --row;
        puts("");
    }
}
```

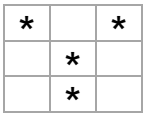
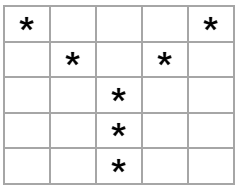
Solution:

The flow chart has been drawn below:



5. b) Write a C program to display following **'Y'** pattern.

[P.T.O]

Sample Input:	Sample Output:
3	
5	

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            if(i > n/2) {
                if(j == n/2) {
                    printf("*");
                }
                else {
                    printf(" ");
                }
            }
            else {
                if(j == i) {
                    printf("*");
                }
                else if(j == n-1-i) {
                    printf("*");
                }
                else {
                    printf(" ");
                }
            }
        }
        printf("\n");
    }
    return 0;
}
```

Summer 2022

1. a) **Rewrite** the following code after **correcting the errors**.

```
include <stdio.h>
void main() {
    int num1 = 5, float num2, char chr = 'q';
    scanf(“%d”, num2);
    num1 = num2 % chr;
    printf("Result is = %f ", num1);
}
```

Solution:

The code has been rewritten below with the error correction:

```
#include <stdio.h>
void main() {
    int num1 = 5;
    float num2;
    char chr = 'q';
    scanf("%f", &num2);
    num1 = (int)num2 % (int)chr;
    printf("Result is = %d ", num1);
}
```

1. b) **Identify** the **invalid variable names** from the following. **Mention the reasons** that make them invalid.

`largest_val, smallest-val, while, 2ndNum, !New, avg marks, val9`

Solution:

The following variable names are invalid from following list:

- `smallest-val`
  Reason: Variable names cannot have hyphens
- `while`
  Reason: Variable names cannot be any keyword
- `2ndNum`
  Reason: Variable names cannot start with number
- `!New`
  Reason: Variable names cannot have exclamatory sign
- `avg marks`
  Reason: Variable names cannot have spaces

1. c) **Compute** the values of the variables **a, b, c,** and **d**.

`float a=5*(5/2), int b=5*(5/2), float c=5*(5.0/2), int d=5*(5.0/2)`

Solution:

The values of the given variables have been written below:

- a = 10.000000
- b = 10
- c = 12.500000
- d = 12

2. a) Write down the **output** of the following C program, for **num = 1** and **num = 3**.

```
#include <stdio.h>
int main() {
    int num;
    int sum = 0, i = 10, j = 5;
    scanf("%d", &num);
    switch(num) {
        case 1:
            sum = 2*i++;
            j++;
        case 2:
            sum = 2*j--;
            i++;
            break;
        case 3:
            sum = ++i*j--;
        case 4:
            sum = i++*j--;
        default:
            sum=0;
            i=0;
            j=0;
    }
    printf("%d %d %d", i, j, sum);
    return 0;
}
```

Solution:

Output for num = 1 :

```
12 5 12
```

Output for num = 3 :

```
0 0 0
```

2. b) **Manually trace** the following code segment and show the **changes of values of the variables *i, sum, b, a, y, x*** in each step.

```
int sum=0, i, a = 1, b, x = 1, y = 1;
for(i=1; i<=5; i++) {
    sum = sum + a;
    b = 6*x + 1;
    a = a + b;
    y++;
    x = x + y;
}
```

Solution:

All the changes of the variables **i**, **sum**, **b**, **a**, **y**, **x** in each step has been shown below:

i	sum	b	a	y	x	i<=5	sum = sum+a	b = 6*x+1	a = a+b	y++	x = x+y
1	0		1	1	1	1<=5 (True)	sum = 0+1 = 1	b = 6*1+1 = 7	a = 1+7 = 8	2	x = 1+2 = 3
2	1	7	8	2	3	2<=5 (True)	sum = 1+8 = 9	b = 6*3+1 = 19	a = 8+19 = 27	3	x = 3+3 = 6
3	9	19	27	3	6	3<=5 (True)	sum = 9+27 = 36	b = 6*6+1 = 37	a = 27+37 = 64	4	x = 6+4 = 10
4	36	37	64	4	10	4<=5 (True)	sum = 36+64 = 100	b = 6*10+1 = 61	a = 64+61 = 125	5	10+5 = 15
5	100	61	125	5	15	5<=5 (True)	sum = 100+125 = 225	b = 6*15+1 = 91	a = 125+91 = 216	6	15+6 = 21
6	225	91	216	6	21	6<=5 (False)					

3. a) **Replace** the nested **"for"** loop in the following code using nested **"do-while"** loop **without changing the logical meaning** of the program:

```
void main() {
    int n = 3, i, j, sum = 0;
    for(i = 0; i < n; i++) {
        for(j = 0; j <= i; j++) {
            if(i == j) sum += i + j;
            else if(i > j) sum += i + n;
            else sum += n - j;
        }
    }
}
```

Solution:

The code has been rewritten below as per the question:

```
int main() {
    int n = 3, i, j, sum = 0;
    i = 0;
    do {
        j = 0;
        do {
            if(i == j) sum += i + j;
            else if(i > j) sum += i + n;
            else sum += n - j;
            j++;
        } while (j <= i);
        i++;
    } while (i < n);
}
```

3. b) **Write a program** to find the **online average** of the **positive numbers** given as inputs by the user. To solve this problem, you should do the following:

i. Write an **infinite loop** that will terminate if the user gives 0 as input.

ii. If the user gives a **positive number** as input, you should keep adding it.
iii. You should also **keep track** of how many positive numbers are given as inputs.
iv. Finally, when the loop terminates, you should **calculate the average** by dividing the sum of the positive numbers by the total positive numbers.

Solution:

The program has been written below:

```
#include <stdio.h>

 int main() {
    int infinite = 1;
    float input;
    int positiveNumber = 0;
    float sum = 0;

    while(infinite == 1) {
        scanf("%f", &input);

        if(input > 0) {
            positiveNumber++;
            sum = sum+input;
        }
        else if(input == 0) {
            infinite = 0;
        }
    }

    float average = sum/positiveNumber;
    printf("Average: %f", average);
}
```

4. a) Show the **manually tracing** (show the values of all the variables and array elements
in each step) for following code segment.

```
int F[6] = {0};
int i, n;
n = 3;
for(i = 0; i<6 ; i++){
    F[i] = n+i;
    if(F[i]%2 == 0){
        F[i] *= 2;
    }
}
```

Solution:

All the changes of the variables in each step has been shown below:

F	n	i	i<6	F[i] = n+i	F[i]%2 == 0	F[i] *= 2
{ 0, Null, Null, Null, Null, Null }	3	0	True	F[0] = 3+0 = 3	False	

{ 3, Null, Null, Null, Null, Null }	"	1	True	F[1] = 3+1 = 4	True	F[1] = 4*2 = 8
{ 3, 8, Null, Null, Null, Null }	"	2	True	F[2] = 3+2 = 5	False	
{ 3, 8, 5, Null, Null, Null }	"	3	True	F[3] = 3+3 = 6	True	F[3] = 6*2 = 12
{ 3, 8, 5, 12, Null, Null }	"	4	True	F[4] = 3+4 = 7	False	
{ 3, 8, 5, 12, 7, Null }	"	5	True	F[5] = 3+5 = 8	True	F[5] = 8*2 = 16
{ 3, 8, 5, 12, 7, 16 }	"	6	False			

4. b) **Write a program** to perform the following operation:

i. Read n **integer numbers** from keyboard and **store** them in an array of size 100, where n is input integer from keyboard.

ii. **Print** all the array elements with their indices (plural of index) in the following format.

```
Index    Value
--------  --------
   0        11
   1        7
   ..       ..
```

iii. **Find and print** the **average** of the numbers that are stored in **odd numbered indices** in the array.

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    int arr[100];

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

    int sum = 0;
    int count = 0;

    printf("Index     Value\n");
    printf("-----     -----\n");
    for(int i=0; i<n; i++){
        printf("%5d     %5d\n", i, arr[i]);

        if(i%2 != 0) {
            sum = sum+arr[i];
```

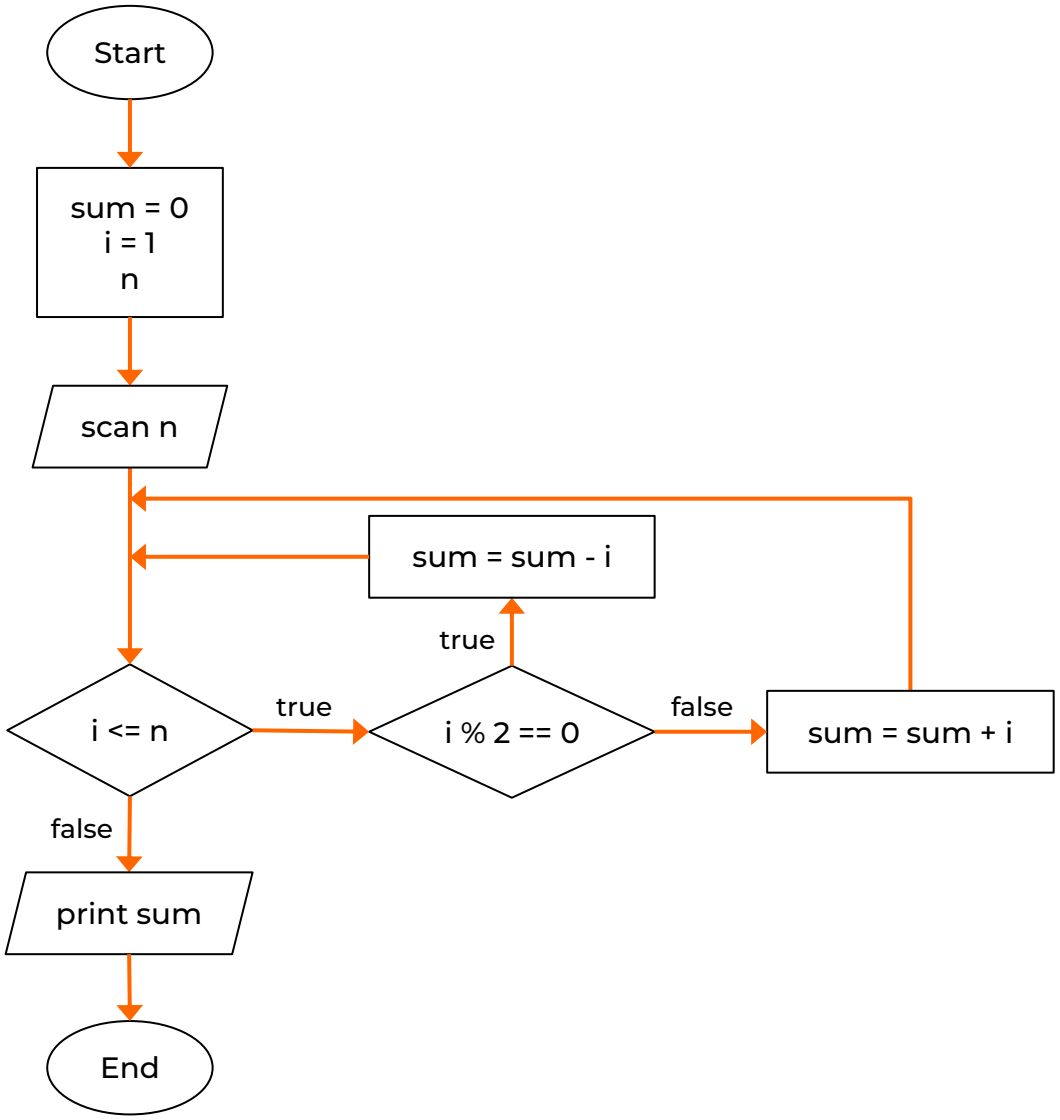
```
            count++;
        }
    }
    int average = sum/count;
    printf("Average: %d\n", average);

    return 0;
}
```

5. a) **Draw a flowchart** to find the **sum** of the following series up to n terms, where n is input integer number from keyboard.

$$1 - 2 + 3 - 4 + \cdots upto\ n\ terms$$

Solution:

The flowchart has been drawn below:



3. a) **Write a program** that takes an integer ***n*** as input from the user and **prints** the following **pattern**. Program for n, NOT 3 or 5.

[P.T.O]

Sample input, n	Sample output
3	6 4 2 4 2 2
5	10 8 6 4 2 8 6 4 2 6 4 2 4 2 2

Solution:

The program has been written below:

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    for(int i=n; i>=1; i--) {
        int count = i*2;
        for(int j=0; j<i; j++) {
            printf("%d   ", count);
            count = count-2;
        }
        printf("\n");
    }

    return 0;
}
```