

PF LAB 06

QUESTION# 01:

Write a program which will find the factorial of a given number. Exit the program if the input number is negative. Also justify your choice of loop answering two important points: why your choice is optimal? Why other looping structure would not be suitable?

Example of Factorial: Input number = 5 Factorial is=5*4*3*2*1

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    int num,factorial=1;
    printf("Enter a number to get Factorial: ");
    scanf("%d",&num);

    if(num<0)
    {
        return 0;
    }
    else
    {
        printf("Factorial is %d!=",num);
        for(int i=num;i>=1;i--)
        {
            factorial*=i;
            if(i>1)
            {
                printf("%d x ",i);
            }
            else
            {
                printf("%d",i);
            }
        }
        printf("= %d ",factorial);
    }
}
```

OUTPUT:

```
Output
Enter a number to get Factorial: 5
Factorial is 5!=5 x 4 x 3 x 2 x 1= 120

=== Code Execution Successful ===
```

Q: why your choice is optimal?

A: The for loop is best suited when you **know exactly how many times** you want to repeat a block of code. The 'for' loop combines initialization (int i = num), condition (i >= 1), and update (i - -) in one line, making it:

- Easier to **read and maintain**.
- **Less error-prone**, since the control logic is in one place.

Q: Why other looping structure would not be suitable?

A: **while loop:**

- A while loop is better for cases where the number of iterations is not known beforehand.
- It separates initialization and update outside the loop, which makes it less compact for this predictable counting process.

Do-while loop:

- A do...while loop executes at least once, even if the condition is false.
- For a factorial, this could cause logical errors if the user enters 0 or a negative number.
- It's therefore not ideal for input-dependent conditions like this one.

QUESTION #02:

Write a program that:

1. Takes a 5-digit number as input.
2. If the sum of its digits is even, check whether the number is prime.
3. If the sum of its digits is odd, check whether the number is a palindrome.

Prime Check: A number is prime if it is divisible only by 1 and itself.

Palindrome Check: A number is a palindrome if it reads the same backward and forward.

SOURCE CODE:

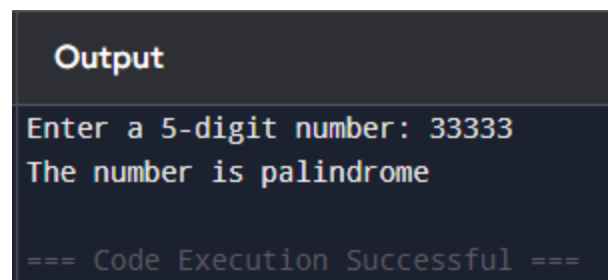
```
#include<stdio.h>
int main()
{
    int num,sum=0,org;

    printf("Enter a 5-digit number: ");
    scanf("%d", &num);
    org=num;
    for(int i = 0; i < 5; i++)
    {
        sum += num % 10;
        num /= 10;
    }
    if(sum%2==0)
    {
        int flag=1;
        for(int i=2;i<=org/2;i++)
        {
            if(org%i==0)
            {
                flag=0;
            }
        }
        if(flag==1)
        {
            printf("Num is prime");
        }
        else
        {
            printf("Num is not prime");
        }
    }
}
```

```
else
{
    int reversed=0,remainder,num1;
    num1=org;
    for(int i=1;i<=5;i++)
    {
        remainder = num1 % 10;
        reversed = reversed * 10 + remainder;
        num1 /= 10;
    }
    if(org==reversed)
    {
        printf("The number is palindrome");
    }
    else
    {
        printf("Not palindrome");
    }
}

return 0;
}
```

OUTPUT:



The screenshot shows a dark-themed window titled "Output". Inside, the text "Enter a 5-digit number: 33333" is displayed on the first line, followed by "The number is palindrome" on the second line. At the bottom, a status message reads "=== Code Execution Successful ===".

QUESTION #03:

Write a program that prints a complex star pattern using both for and while loops. Below is an example of the expected output:

```

      *
    ***
  *****
*****
*****
  *****
    ***
      *

```

The number of rows should be adjustable based on user input (must be an odd number). The pattern should be printed using both for and while loops, and the program should compare the efficiency of both approaches.

SOURCE CODE:

FOR LOOP:

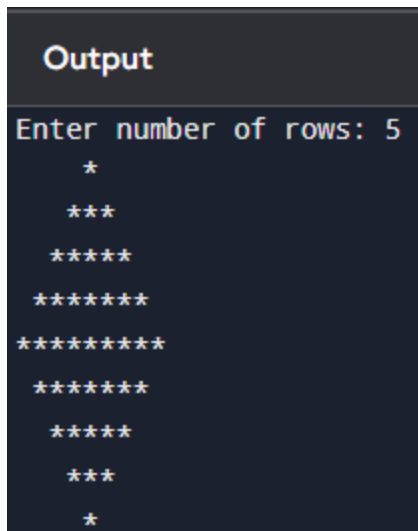
```

#include<stdio.h>
int main()
{
    int i,j,rows;
    printf("Enter number of rows: ");
    scanf("%d",&rows);
    for(i=1;i<=rows;i++)
    {
        for(j=1;j<=rows-i;j++)
        {
            printf(" ");
        }
        for(j=1;j<=2*i-1;j++)
        {
            printf("*");
        }
        printf("\n");
    }
    for(i=rows-1;i>=1;i--)
    {
        for(j=1;j<=rows-i;j++)
        {
            printf(" ");

```

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

OUTPUT:



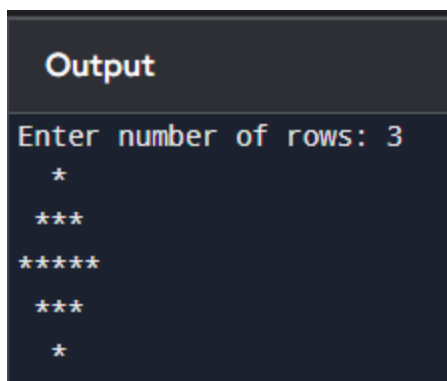
```
Output  
Enter number of rows: 5  
 *  
***  
*****  
*****  
*****  
*****  
*****  
***  
 *
```

WHILE LOOP:

```
#include<stdio.h>  
int main()  
{  
    int rows;  
    printf("Enter number of rows: ");  
    scanf("%d",&rows);  
    int i,j;  
    i=1;  
    while(i<=rows)  
    {  
        j=1;  
        while(j<=rows-i)  
        {  
            printf(" ");  
            j++;  
        }  
    }
```

```
j=1;
while(j<=2*i-1)
{
    printf("*");
    j++;
}
printf("\n");
i++;
}
i=rows-1;
while(i>=1)
{
    j=1;
    while(j<=rows-i)
    {
        printf(" ");
        j++;
    }
    j=1;
    while(j<=2*i-1)
    {
        printf("*");
        j++;
    }
    printf("\n");
    i--;
}
return 0;
}
```

OUTPUT:



```
Output
Enter number of rows: 3
 *
***
*****
***
 *
```

QUESTION #04:

Develop a user registration system with the following features:

1. Ask for a username of 5 alphabets.
2. Ask for a password that should be:
 - 6 characters long,
 - Contain at least 1 numeric digit,
 - Contain at least 1 uppercase letter,
 - Contain at least 1 lowercase letter.
3. If both the username and password are valid, display: "Account Created Successfully".
4. Prompt the user to log in using the correct username and password. If login is successful, display: "Welcome username, you are now logged in."

The system should use both break and continue statements to validate input.

SOURCE CODE:

```
#include<stdio.h>
#include<string.h>
int main()
{
    char username[6],password[7],user2[6],pass2[7];
    int len;
    printf("Enter username of 5 alphabet: ");
    scanf("%s",username);
    len=strlen(username);
    while(1){
        if(len!=5)
        {
            printf("Invalid username\n");
            scanf("%s",username);
            len=strlen(username);
            continue;
        }
        else
        {
            break;
        }
    }
    int digit=0,upper=0,lower=0;
    printf("Create password with");
    printf("\n6 characters long");
    printf("\nContain at least 1 numeric digit");
    printf("\nContain at least 1 uppercase letter");
```



```
printf("\nContain at least 1 lowercase letter\n");
while(1)
{
    int digit = 0, upper = 0, lower = 0;

    printf("\nEnter password: ");
    scanf("%s", password);
    len = strlen(password);

    if (len != 6)
    {
        printf("Invalid: Password must be exactly 6 characters long.\n");
        continue;
    }
    for (int i = 0; i < len; i++)
    {
        if (password[i] >= '0' && password[i] <= '9')
            digit = 1;
        if (password[i] >= 'A' && password[i] <= 'Z')
            upper = 1;
        if (password[i] >= 'a' && password[i] <= 'z')
            lower = 1;
    }

    if (digit && upper && lower)
    {
        printf("Account Created Successfully.\n");
        break;
    }
    else
    {
        printf("Invalid password. Must contain: ");
        if (!digit)
            printf("a digit ");
        if (!upper)
            printf("an uppercase letter ");
        if (!lower)
            printf("a lowercase letter");
        printf("\nPlease try again.\n");
    }
}
```

```
printf("---Login to your Account---");
int cmp1,cmp2;
while(1){
printf("\nEnter Username: ");
scanf("%s",user2);
cmp1=strcmp(username,user2);
if(cmp1!=0)
{
printf("Incorrect Username");
cmp1=strcmp(password,pass2);
}
else
{
break;
}
}
while(1){
printf("Enter Password: ");
scanf("%s",pass2);
cmp2=strcmp(password,pass2);
if(cmp2!=0)
{
printf("Incorrect Password\n");
continue;
}
else
{
break;
}
}
printf("Welcome %s, you are now logged in",username);
}
```

OUTPUT:

```
Output
Enter username of 5 alphabet: aadil
Create password with
6 characters long
Contain at least 1 numeric digit
Contain at least 1 uppercase letter
Contain at least 1 lowercase letter

Enter password: Aadil1
Account Created Successfully.
---Login to your Account---
Enter Username: aadil
Enter Password: Aadil1
Welcome aadil, you are now logged in
```

QUESTION #05:

Write a program that generates a Fibonacci triangle where each row displays a Fibonacci number. The number of rows should be specified by the user.

Example for n = 5:

```
1
1 1
2 3 5
8 13 21 34
55 89 144 233 377
```

The program should:

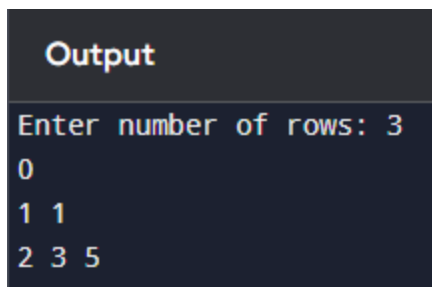
1. Use a loop to print the Fibonacci numbers in a triangular format.
2. Use nested loops to print each row dynamically.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    int i,j,row,a=0,b=1,c;
    printf("Enter number of rows: ");
    scanf("%d",&row);
    for(i=1;i<=row;i++)
    {
```

```
    for(j=1;j<=i;j++)
    {
        printf("%d ",a);
        c = a + b;
        a = b;
        b = c;
    }
    printf("\n");
}
```

OUTPUT:



```
Output
Enter number of rows: 3
0
1 1
2 3 5
```

QUESTION #06:

Write a program that prints a pyramid-shaped pattern of numbers, where the top row starts with 1 and increases sequentially to the center. Then, the numbers decrease symmetrically. The number of rows is entered by the user.

For n = 5, the pattern would look like this:

```
  1
 121
12321
1234321
123454321
```

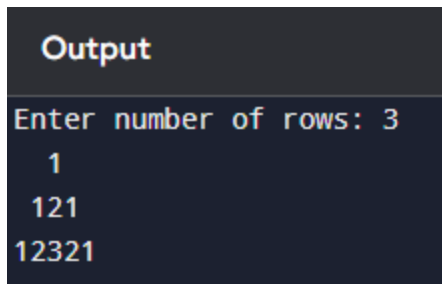
SOURCE CODE:

```
#include<stdio.h>
int main()
{
    int i,j,row,a=1,b=0,c;
    printf("Enter number of rows: ");
    scanf("%d",&row);
    for(i=1;i<=row;i++)
    {
        for(j=row;j>i;j--)
```

```
{
    printf(" ");
}
for (int j = 1; j <= i; j++)
{
    printf("%d", j);
}
for (int j = i - 1; j >= 1; j--)
{
    printf("%d", j);
}

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

OUTPUT:



```
Output
Enter number of rows: 3
1
121
12321
```

QUESTION #07:

Write a program that:

1. Accepts a sentence from the user.
2. Reverses the order of the words in the sentence, maintaining the order of characters within each word.
3. The program should process the sentence using a loop without using any advanced data structures (like arrays or strings).

Example:

Input: "I love programming"

Output: "programming love I"

SOURCE CODE:

```
#include <stdio.h>
int main()
{
    char arr[100];
```

```
int count = 0;
char ch;
printf("Enter a sentence: ");
while (count < 100)
{
    ch = getchar();
    if (ch == '\n')
        break;
    arr[count++] = ch;
}
arr[count] = '\0';
for (int i = 0; i < count / 2; i++)
{
    char temp = arr[i];
    arr[i] = arr[count - i - 1];
    arr[count - i - 1] = temp;
}
int start = 0;
for (int i = 0; i <= count; i++)
{
    if (arr[i] == ' ' || arr[i] == '\0')
    {
        int end = i - 1;
        while (start < end)
        {
            char temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
        start = i + 1;
    }
}

printf("\nReversed word order: %s\n", arr);
return 0;
}
```

OUTPUT:

Output

```
Enter a sentence: I love Programming  
Reversed word order: Programming love I
```

QUESTION #08:

Write a program that takes Input a number and check if it's an Armstrong number using a while loop.

Example: 153 is Armstrong because $1^3 + 5^3 + 3^3 = 153$.

SOURCE CODE:

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

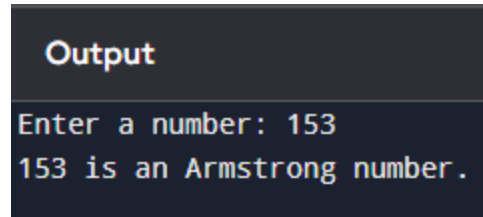
int main()
{
    int num,orgnum,remainder,result=0, count=0;

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

    orgnum=num;
    int temp=num;
    while (temp != 0)
    {
        temp /= 10;
        count++;
    }
    temp=num;
    while (temp != 0)
    {
        remainder=temp % 10;
        result += pow(remainder, count);
        temp/= 10;
    }
    if (result==orgnum)
        printf("%d is an Armstrong number.\n", orgnum);
    else
        printf("%d is not an Armstrong number.\n", orgnum);
}
```

```
    return 0;  
}
```

OUTPUT:



```
Output  
Enter a number: 153  
153 is an Armstrong number.
```

QUESTION #09:

Input a number, find how many digits it has, then display each digit separately in order (from first to last).

Example:

Input: 4321

Output:

Digit 1: 4

Digit 2: 3

Digit 3: 2

Digit 4: 1

Use Do While loop

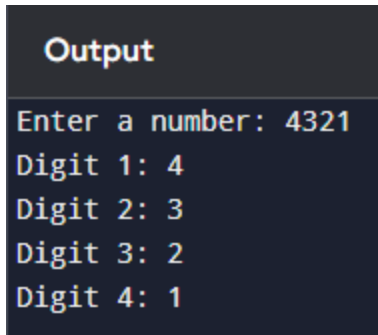
SOURCE CODE:

```
#include<stdio.h>  
int main()  
{  
    int num,temp,count=0,digit,reversed=0;  
    printf("Enter a number: ");  
    scanf("%d",&num);  
    temp=num;  
    while(temp != 0)  
    {  
        digit=temp%10;  
        reversed=reversed*10+digit;  
        temp/=10;  
        count++;  
    }  
    int i=1;  
    do  
    {
```



```
    digit=reversed%10;
    printf("Digit %d: %d\n",i,digit);
    reversed/=10;
    i++;
}
while(i<=count);
return 0;
}
```

OUTPUT:



```
Output
Enter a number: 4321
Digit 1: 4
Digit 2: 3
Digit 3: 2
Digit 4: 1
```

QUESTION #10:

Input a number and calculate the sum of digits that are in odd positions when counted from right to left (1st, 3rd, 5th, etc.) using a loop.

Example:

Input: 12345

Digits from right: 5 (pos 1), 4 (pos 2), 3 (pos 3), 2 (pos 4), 1 (pos 5)

Sum of digits in odd positions = $5 + 3 + 1 = 9$

SOURCE CODE:

```
#include<stdio.h>
int main()
{
    int num,temp,count=0,digit,reversed=0,sum=0,temp2;
    printf("Enter a number: ");
    scanf("%d",&num);
    temp=num;temp2=num;
    while(temp != 0)
    {
```

```
        temp/=10;
        count++;
    }
    int i=1;
    printf("Digits from right: ");
    do
    {
        digit=num%10;
        printf("%d(pos %d), ",digit,i);
        num/=10;
        i++;
    }
    while(i<=count);
    printf("\nSum of digits in odd position= ");
    int digit2,j=1;
    while(temp2 != 0)
    {
        digit2 = temp2 % 10;
        printf("%d + ",digit2);
        sum += digit2;
        temp2 /= 100;
    }
    printf("\b= %d",sum);

    return 0;
}
```

OUTPUT:

Output

Clear

Enter a number: 12345

Digits from right: 5(pos 1), 4(pos 2), 3(pos 3), 2(pos 4), 1(pos 5),

Sum of digits in odd position= 5 + 3 + 1 + = 9