Practical Workbook

CT-175 PROGRAMMING FUNDAMENTALS

Name: Hamna Ali Khan

Year: 2024

Batch: 2028

Roll No: CT-157

Department: BCIT

Dept. of Computer Science & Information Technology

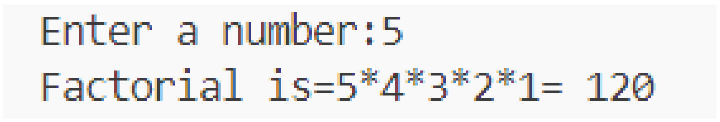
NED University of Engineering & Technology

EXERCISE Q# 01

Write a program which will find the factorial of a given number. Exit the program if the input number is negative. Test case: Input number = 5,Factorial is=5*4*3*2*1

```c
#include<stdio.h>
int main(){
    int a, fact=1;
    printf("Enter a number:");
    scanf("%d", &a);
    if (a<0){
        printf("Invalid. The factorial of a negative number is not possible");
        return 1;
    }
    printf("Factorial is=");
    for(int i=a;i>=1;i--){
        fact*=i;
        printf("%d",i);
        if(i>1){
        printf("*");
        }
    }

    printf("= %d\n", fact);
}
```

OUTPUT:



```
Enter a number:5
Factorial is=5*4*3*2*1= 120
```

EXERCISE Q# 02

Write a program which will generate the Fibonacci series up to 1000. Also find the sum of the generated Fibonacci numbers divisible by 3, 5 and 7 only.
Fibonacci series is:1 1 2 3 5 8 13 25..........
Note: Do this task by using ***for loop*** and ***while loop***. Also identify which one is more efficient?

WHILE-LOOP:

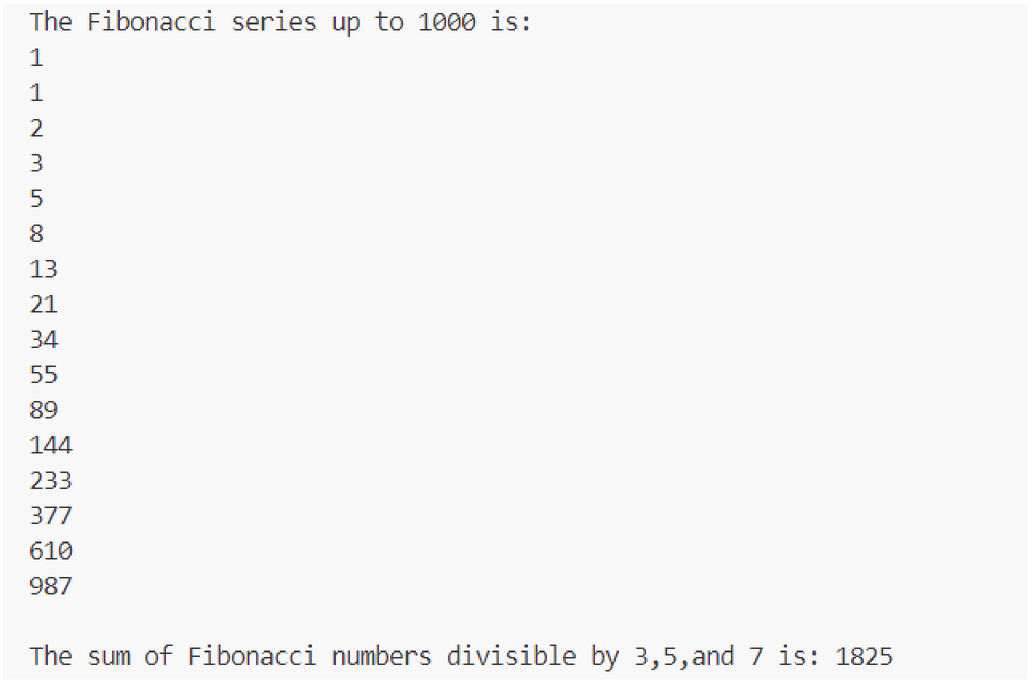
```
#include <stdio.h>
int main() {
    int a = 1, b = 1, sum = 0;
    int thirdterm = a + b;
    printf("The Fibonacci series up to 1000 is:\n%d\n%d\n", a, b);

    while (thirdterm <= 1000) {
        printf("%d\n", thirdterm);
        if (thirdterm % 3 == 0 || thirdterm % 5 == 0 || thirdterm % 7 == 0) {
            sum += thirdterm;
        }
        a = b;
        b = thirdterm;
        thirdterm = a + b;
    }

    printf("\nThe sum of Fibonacci numbers divisible by 3,5,and 7 is: %d\n",
sum);

    return 0;
}
```

FOR LOOP:

```
#include <stdio.h>
int main() {
    int a=1,b=1,sum=0;
    int thirdterm = a+b;
    printf("The Fibonacci series up to 1000 is:\n%d\n%d\n",a,b);
    for (; thirdterm <= 1000; a=b, b= thirdterm, thirdterm =a+b) {
        printf("%d\n",thirdterm);

        if (thirdterm%3==0 || thirdterm%5==0 || thirdterm%7==0) {
            sum += thirdterm;
        }
    }
    printf("\nThe sum of Fibonacci numbers divisible by 3, 5,and 7 is: %d\n",
sum);
    return 0;
}
```

CONCLUSION:
The while loop is more efficient because it continues until Fibonacci values exceed 1000, making it ideal for condition-based checks. In contrast, a for loop works best with an initial value and a specific number of iterations. Since multiple variables need updates each time, the while loop is the better option here.

OUTPUT:



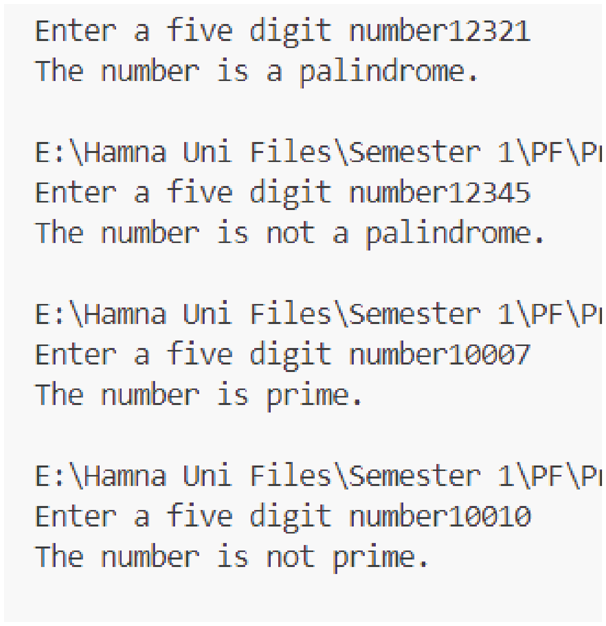
```
The Fibonacci series up to 1000 is:
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987

The sum of Fibonacci numbers divisible by 3,5,and 7 is: 1825
```

EXERCISE Q# 03

Write a program which will input a 5-digit number. If the sum of digits is even, find whether the input number is a prime or not. If the sum of digits is odd find, whether the number is palindrome or not?
Example of prime number: A number which is only divisible by itself and 1 i.e., 7, 11, and13.
Example of a Palindrome: A number whose reverse order is the same as the original number i.e., 11211, 44344.

```
#include<stdio.h>
int main(){
    int num,sum=0;
    printf("Enter a five digit number");
    scanf("%d", &num);
    int originalnum;
    originalnum=num;
int a, reverse=0;
while(num>0){
  a=num%10;
    sum+=a;
    reverse=reverse*10+a;
    num=num/10;
}
if (sum % 2 == 0) {
        if (originalnum > 1) //because 1 and 0 are not prime
        {
            int i;
            for (i = 2; i <= originalnum / 2; i++) {
                if (originalnum % i == 0) {
                    printf("The number is not prime.\n");
                    break;
                }
            }

            if (i > originalnum / 2) {
                printf("The number is prime.\n");
            }
        } else {
            printf("The number is not prime.\n");
        }
    } else {

        if (originalnum == reverse) {
            printf("The number is a palindrome.\n");
        } else {
            printf("The number is not a palindrome.\n");
        }
    }
  return 0;
}
```

OUTPUT:



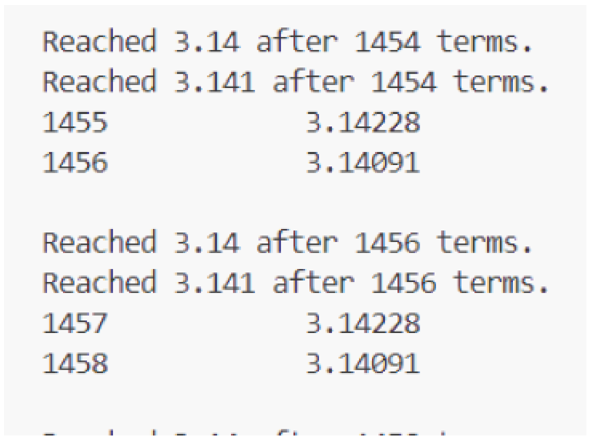
```
Enter a five digit number12321
The number is a palindrome.

E:\Hamna Uni Files\Semester 1\PF\Pı
Enter a five digit number12345
The number is not a palindrome.

E:\Hamna Uni Files\Semester 1\PF\Pı
Enter a five digit number10007
The number is prime.

E:\Hamna Uni Files\Semester 1\PF\Pı
Enter a five digit number10010
The number is not prime.
```

EXERCISE Q# 04

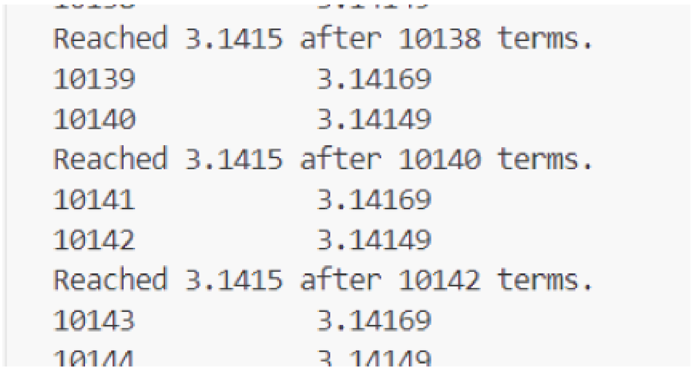
(Calculating the Value of π) Calculate the value of π from the infinite series. Print a table that shows the value of π approximated by one term of this series, by two terms, by three terms, and so on. How many terms of this series do you have to use before you first get 3.14? 3.141? 3.1415? 3.14159?

```
#include <stdio.h>
int main() {
double pi = 0.0;
int sign = 1;
int terms = 0;
printf("Terms\t\tπ Approximation\n");
printf("-----------------------------------------------\n");
for (int i = 0; ; i++) {
pi += sign * (4.0 / (2 * i + 1));
sign *= -1;
terms++;
printf("%d\t\t%f\n", terms, pi);
if (pi > 3.14 - 0.001 && pi < 3.14 + 0.001) //0.001 to capture values slightly
below or above 3.14 for pi approximation
{
printf("\nReached 3.14 after %d terms.\n", terms);
}
if (pi > 3.141 - 0.0001 && pi < 3.141 + 0.0001) //0.0001 to capture values
slightly below or above 3.141 for pi approximation
{
printf("Reached 3.141 after %d terms.\n", terms);
}
if (pi > 3.1415 - 0.00001 && pi < 3.1415 + 0.00001) //0.00001 to capture values
slightly below or above 3.1415 for pi approximation
{
printf("Reached 3.1415 after %d terms.\n", terms);
}
if (pi > 3.14159 - 0.000001 && pi < 3.14159 + 0.000001) //0.000001 to capture
values slightly below or above 3.14159 for pi approximation
{
printf("Reached 3.14159 after %d terms.\n", terms);
break;
}
      }
  return 0;
}
```

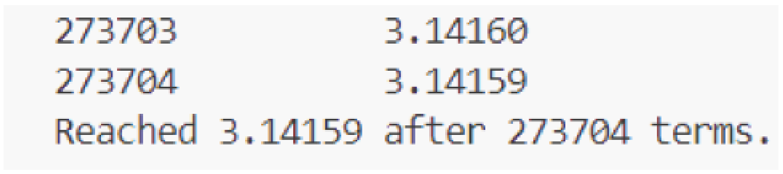
OUTPUT:



```
Reached 3.14 after 1454 terms.
Reached 3.141 after 1454 terms.
1455            3.14228
1456            3.14091

Reached 3.14 after 1456 terms.
Reached 3.141 after 1456 terms.
1457            3.14228
1458            3.14091
```



```
Reached 3.1415 after 10138 terms.
10139           3.14169
10140           3.14149
Reached 3.1415 after 10140 terms.
10141           3.14169
10142           3.14149
Reached 3.1415 after 10142 terms.
10143           3.14169
10144           3.14149
```



```
273703          3.14160
273704          3.14159
Reached 3.14159 after 273704 terms.
```