سوال ۱

جواب اخر :

This is case 0 This is case 1

توضیحات :

در گام اول کد وارد حلقه می شود و بعد وارد به روی متغیر i سوییچ میشود در خط بعد یک statement تعریف شده که مقدار i را از ۰ به ۲ می خواهد تغییر دهد اما چون در دستور switch/case ما نمیتوانیم statement تعریف کنیم کامپایلر آن را در نظر نمیگیرد و با i=0 به کار خود ادامه می دهیم و وارد case=0 میشویم و پرینت می کند This is case 0 و به مقدار i یکی اضافه میکنیم و به خاطر اینکه دستور break استفاده نشده وارد case=1 میشود و پرینت می کند This is case 1 و دوباره به مقدار i یکی اضافه میکنیم و بعد دستور break باعث میشود از switch خارج شویم و به خاطر اینکه مقدار i دیگر کمتر ۲ نیست از حلقه هم خارج میشود که در نهایت خروجی زیر در ترمینال پرینت میشود :

This is case 0 This is case 1

سوال ۲ :

```c
#include <stdio.h>
int main()
{
    int x = 1;
    while (x <= 10)
    {
        ++x;
    }
    for (double y = .1; y <= 1.0; y += .1)
    {
        printf("%f\n", y);
    }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
padir@padir-Legion-5-16IRX9:~/Documents/c_codes/codes$ ./a.out
0.100000
0.200000
0.300000
0.400000
0.500000
0.600000
0.700000
0.800000
0.900000
1.000000
```

سوال ۳ :

```
// question 3
fixed_z=FLOAT INPUT
z=fixed_z
n=INT INPUT // starting n from 1
n=n-1 // beacuse we want to start n from 0 not 1
i=0
final_answer=0
/* we could write factorial with recursive function instead what just we did*/
FUNCTION factorial(INT num)
    answer=1
    FOR num to 2 with steps -1 // for(num ; num>1 ;num-=1 )
        answer=answer*num
    ENDFOR
    return answer
END FUNCTION
// i assume ^ means power for example 2^4=16
FOR i to n with steps 1 // for(i;i<=n ; i+=1)
    z=fixed_z
    z=((-1^i)(z^(2i+1)))/(factorial(2i+1))
    final_answer = final_answer + z
ENDFOR
print final_answer
```

سوال 4 :

```
//question 4
a= INT INPUT
b= INT INPUT
a= a % 10
b= b % 10
FUNCTION is_relative_prime(INT num1 , INT num2) // defining function
    biggest = 0
    smallest_num=0
    relative_prime_flag=1 // means true
    IF num1 > num2 THEN
        biggest=num1
        smallest_num=num2
    ELSE
        biggest=num2
        smallest_num=num1
    ENDIF
    smallest=smallest_num
    i=1

    FOR smallest to i with steps -1  // for (smallest;smallest>= i ; smallest-=1)
        if smallest_num%smallest=0 AND biggest%smallest=0 THEN
            relative_prime_flag=0 // means false
            break
        ENDIF
    ENDFOR
    return relative_prime_flag
END FUNCTION
IF is_relative_prime(a,b) = 1 THEN
    print YES
ELSE
    print NO
ENDIF
```

سوال ۵ بخش الف :

```
64  //question 5 part A
65  number = INT INPUT
66  num_count_fix=3 // pre defined by problem
67  num_count = num_count_fix
68  // i assume ^ means power for example 2^4=16
69  answer = 0
70  FOR num_count to 1 with steps -1 // for(num_count;num_count>0;num_count-=1)
71      digit=0
72      exponant_pow = num_count-1
73      digit=(number/(10^exponant_pow))%10
74      answer = answer + (digit*(10^(num_count_fix-exponant_pow-1)))
75  print answer
```

سوال ۵ بخش ب :

```c
#include <stdio.h>
int power(int base , int exponant) {
   int answer = 1 ;
   for (exponant;exponant>0 ; exponant-=1)
   {
       answer=answer*base ;
   }
   return answer ;
}
int main()
{
   int num ;
   scanf("%d",&num) ;
   if (num%10==0)
   {
       printf("this number can be reverse because it has 0 at last\n");
       return -1 ;
   }
   int digits_count = 3 ; //assumed by problem
   int fix_digits_count = digits_count ;
   int out_put=0 ;
   int each_digit , exponant ;
   for (digits_count ; digits_count>0 ; digits_count-=1)
   {
       exponant=digits_count-1 ;
       each_digit = num/(power(10,exponant))%10;

       out_put+= (each_digit*power(10,fix_digits_count-exponant-1));
   }
   printf("%d\n",out_put);
   return 0 ;
}
```

سوال ۶ :

```c
//question 6
#include <stdio.h>
#include <stdbool.h>
bool is_prime_num(int num)
{
   if(num<=1) return false ;
   bool prime_flag = true ;
   /*there is more efficient way and for doing that we have to sqrt the
num and we can do this instead what actually we did
   for (int i=2 ;i<sqrt(num) ; i+=1 )
   and there is math proof behind it
   */
   for (int i=2 ;i<num ; i+=1 )
   {
       if (num%i==0)
       {
           prime_flag =false ;
           break;
       }
   }
   return prime_flag ;
}
void main(void)
{
   int g=0 ;
   int start_num , finish_num ; //(start,finish]
   scanf("%d %d",&start_num,&finish_num);
   start_num+=1 ; //[start,finish]
   for (start_num ;start_num<=finish_num ; start_num+=1 )
   {
       if(is_prime_num(start_num))
       {
           g+=1;
           printf("%d ",start_num);
       }
   }
   if (g==0) printf("There are no prime numbers") ;
   printf("\n");
}
```