AUTOMATA FIX Questions

1) Check for syntax error/ logical error and correct the error to get the desired output.
Given n, print from n to 0

```
int main()
{
   int n;
   scanf(“%d”, &n);
   unsigned int i = n;
   while(i >= 0)
   {
          printf(“%d\n”, i);
          i–;
   }
  return 0;
}
```

Input: 4
Output: Infinite loop

Answer: Error – Logical error
unsigned int i = n; unsigned integer ranges from 0 to 65535, which will be taken in the cyclic order. So i– will keep repeating in a cyclic way. The loop will never be terminated. So it should be written as **int i = n;**

2) Find the factorial of a given number.

```
int main()
{
long int fact = 1, n, i;
scanf(“%d”, &n);

for(i =1; i <= n; i++)
{
fact = fact * i;
}
printf(“%d”, fact);
return 0;
}
```

Input: 20
Output: -2102132736

Answer: Error – Logical error
The fact and n are declared as long int, so in scanf and printf %ld should be used in place of %d.

3) Check whether the below program print the below pattern

1111
222
33

```
void main()
{
   int i, j, n;
   scanf("%d", &n);
   for(i = 1; i<n; i++)
   {
      for(j = 1; j<n; j++)
      {
            printf("%d",  i);
      }
     printf("\n");
   }
}
```

Input: 3
Output:
111
222
333

Answer: Error: Logical error
The inner for loop has to be written in this way: **for(j = i-1; j<n; j++)**

4) Find the greatest of three numbers.

```
int main()
{
  int num1, num2, num3;
  scanf("%d %d %d", &num1,&num2,&num3);
  if (num1 > num2) && (num1 > num3)
  {
            printf("%d", num1);
  }
  elseif(num2>num3)
  {
            printf("%d", num2)
  }
  else
  {
            printf("%d", num3);
  }
  return 0;
}
```

Answer: Error: Syntax error
if (num1 > num2) && (num1 > num3) à it has to be written as if ((num1 > num2) && (num1 > num3)) and this line **elseif**(num2>num3) should be rewritten as **else if**(num2>num3)

5) Fix the error, recompile and match against the output provided.

```
int main(void)
{
printf(“This is a \”buggy” program\n”);
   return 0;
}
```

Corrected program:

```
int main(void)
{
   printf(“This is a \”buggy\” program\n”);
   return 0;
}
```

6) Code reuse: Convert Binary to Decimal by using the existing function.

```
void  binarytodecimal(number)
{
          // Type your code here
}
void main()
{
   int num;
   scanf(“%d”, &num);
   printf(“%d”, binarytodecimal(num);
}
```

Answer:

```
void  binarytodecimal(number)
{
 int dval=0, base=1, rem;
 while(number > 0)
 {
          rem = number % 10;
          dval = dval + rem * base;
             num = number / 10;
             base = base * 2;
 }
 return dval;
}
```

7) Print the prime numbers from an array up to given value n by using existing function.

```
int isprime(int num)
```

```
{
// type your code here
}
int main()
{
int n, m, arr[100], size=0, i;
scanf(“%d”, &n);
for(m = 2; m <= n; m++)
{
if(isprime(m))
{
   arr[size++]= m;
}
}
for(i = 0; i < size; i++)
{
printf(“%d\n”, arr[i]);
}
return 0;
}
```

Answer:

```
int isprime(int num)
{
  int i;
  int isprime = 1;
  for(i = 2;  i <= num / 2; i++)
  {
          if(num % i == 0)
          {
                    isprime = 0;
                        break;
          }
   }
   return isprime;
 }
```

8. Find the syntax error in the below code without modifying the logic.

```
#include
int main()
{
   float x = 1.1;
   switch (x)
   {
      case 1: printf("Choice is 1");
               break;
      default: printf("Invalid choice");
              break;
   }
```

```
  return 0;
}
```
Answer:

```
#include
int main()
{
  int x = 1;
  switch (x)
  {
     case 1: printf("Choice is 1");
              break;
     default: printf("Invalid choice");
             break;
  }
  return 0;
}
```
The expression used in the switch must be an integral type (int, char, and enum). Any other type of expression is not allowed.

9. Find the logical error in the below code.

```
void main () {
  int i, j, n = 5;
  for(i=1; i<n; i++)
  {
for(j=i;j<n;j++);
{
       printf("%d", i);
}
printf("\n");
  }
}
```
Solution:
```
void main () {

  int i, j, n = 5;

  for(i=1; i<n; i++)

  {

            for(j=i;j<n;j++)

{
```

```
printf("%d", i);

   }

    printf("\n");

  }

}
```

we use a semicolon in C statement to tell the compiler where's the end of our statement. Second for loop executes one time.

10. Complete the below code by reusing the existing function.

Find the index of equilibrium element in the given array. In an array equilibrium element is the one where the sum of all the elements to the left side is equal to the sum of all the elements in the right side.

Return Value:

1) Return -1 if no equilibrium element is found

2) In case there is more than one equilibrium element, return the element with least index value.

You are required to complete the given code by reusing the existing function. You can click on Compile & run anytime to check the compilation/execution status of the program you can use printf to debug your code. The submitted code should be logically/syntactically correct and pass all the test cases.

Code approach For the question:

You will need to correct the given implementation.
We do not expect you to modify the approach or incorporate any additional library methods.

Test Case:

a[] = {1,2,3,4,3,3}. 4 is the equilibrium element since its left side sum (1+2+3) is equal to its right side sum (3+3)

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

```
// Return the sum of elements from index 0 to (idx - 1)
int left_side_sum(int a[], int n, int idx)
{
    int sum = 0, i;
    for(i = 0; i < idx; i++)
    {
        sum += a[i];
    }
    return sum;
}
// Return the sum of elements from index (idx + 1) to (n - 1)
int right_side_sum(int a[], int n, int idx)
{
    int sum = 0, i;
    for(i = idx + 1; i < n; i++)
    {
        sum += a[i];
    }

    return sum;
```

```
}
// returns -1 if no equilibrium index found
int findEquilibriumIndex(int a[], int n)
{
    // Type your code here
}
int main() {
            //code
   int a[10], n, i;
   // get the elements count
   scanf("%d", &n);
   // get the array elements
   for(i=0; i<n; i++)
   {
      scanf("%d", &a[i]);
   }
   int equiIndex = findEquilibriumIndex(a, n);
   if(equiIndex != -1) {
      printf("%d", a[equiIndex]);
   }
```

```
   return 0;
}
```

Solution:

```
// Return the sum of elements from index 0 to (idx - 1)
int left_side_sum(int a[], int n, int idx)
{
    int sum = 0, i;
    for(i = 0; i < idx; i++)
    {
        sum += a[i];
    }

    return sum;
}

// Return the sum of elements from index (idx + 1) to (n - 1)
int right_side_sum(int a[], int n, int idx)
{
    int sum = 0, i;
```

```
    for(i = idx + 1; i < n; i++)
    {
        sum += a[i];
    }

    return sum;
}

// returns -1 if no equilibrium index found
int findEquilibriumIndex(int a[], int n)
{
    // Type your code here
    int i;
    for(i = 0; i < n; i++)
    {
        if(left_side_sum(a, n, i) == right_side_sum(a, n, i))
        {
            return i;
        }
    }
```

```
    return -1;
}

int main() {
            //code
   int a[10], n, i;
   // get the elements count
   scanf("%d", &n);
   // get the array elements
   for(i=0; i<n; i++)
   {
      scanf("%d", &a[i]);
   }

   int equiIndex = findEquilibriumIndex(a, n);
   if(equiIndex != -1) {
      printf("%d", a[equiIndex]);
   }
   return 0;
```

}