“AUTOMATA CODING ASSIGNMENT”

01 “C# code for Deterministic Finite Automata”

```
namespace Deterministic_Finite_Automata
{
   class node
            {
            public node next = null;
            public node previous = null;        public
            string value;
            public node(node next, node previous, string value)
      {
            this.next = next;
            this.value = value;
            this.previous = previous;
      }
            }
   class Program
   {

            public static node head;
            public static node Head;
            public static int index = 0;
            static void Main(string[] args)
      {
            Console.WriteLine("deterministic finite automata which accept a string containing “the” anywhere
in a string\n");
            Console.WriteLine ("Enter a string");
            string str = Console.ReadLine();
            node previous;
            node next;
             head = new node(null, null, "t");           previous
            = head;
             Head = head;
            next = new node(null, previous, "h");
            previous.next = next;
            previous = next;
            next = new node(null, previous, "e");
            previous.next = next;
            previous = next;
            next = new node(null, previous, "final");
            previous.next = next;
           StateOne (str, head.value);
      }
      public static void StateOne(string str, string value)
      {
            int i;          for
```

```
(i = index; i < str.Length
&& index <
str.Length;i++)
        {               index = i;
           if (str[i].ToString() == value)
           {
               head = head.next;
                index++;
              StateTwo(str, head.value);
           }
        }

        index = i;
     }
     public static void StateTwo(string str, string value)
     {          int i;
        for (i = index; i < str.Length && index < str.Length; i++)
        {
           if (str[i].ToString() == value)
           {
              head = head.next;                  index++;
              StateThree(str, head.value);
           }
           else if (str[i]!='t')
           {
              head = head.previous;
              index++;
              StateOne(str,head.value);
           }
        }
     }
     public static void StateThree(string str, string value)
     {          int i;
        for (i = index; i < str.Length && index < str.Length; i++)
        {
           if(str[i].ToString() == value)
           {
              Console.WriteLine("The word is in the language");
               index = str.Length;
               break;
            }
           else if(str[i] == 't')
           {
              head = head.previous;
              index++;
              StateTwo(str,head.value);
           }
else
        {
```

```
            head = Head;
            index++;
           StateOne(str, head.value);
          }
         }
      }
   }
}
```

---

02 “C# code for Non Deterministic Finite Automata”

```
namespace NonDeterministicFiniteAutomata
{
   class node    {
   public node next;
   public string data;
   public node(node next,string data)
      {
         this.next = next;
         this.data = data;
      }
}
   class Program
   {
      public static int index = 0;
      public static node head;
      public static bool flag = false;
      static void Main(string[] args)
      {
          Console.WriteLine("Non-deterministic finite automata which accept a string containing “the” anywhere
in a string ");
            Console.WriteLine("Enter a string");
            string str = Console.ReadLine();
            node next;
            node previous;
            head = new node(null, "t");
            previous = head;
            next = new node(null, "h");
            previous.next = next;
            previous = next;
            next = new node(null, "e");
            previous.next = next;
            previous = next;
            next = new node(null,"final");
            previous.next = next;
            StateOne(str, head.data);
      }
      public static void StateOne(string str,string value)
      {          int i;
```

```
        for (i = index; i < str.Length && index < str.Length; i++)
        {
            if (str[i].ToString() == value)
            {
                head = head.next;
                  index = ++i;
                StateTwo(str, head.data);
            }
        }
    }
    public static void StateTwo(string str, string value)
    {
        for (int i = index; i < str.Length && flag != true; i++)
        {
            if (str[i].ToString() == value)
            {
                head = head.next;
                  index = ++i;
                StateThree(str, head.data);
            }
else              {
                flag = true;
            }
        }
    }
    public static void StateThree(string str, string value)
    {
        for (int i = index; i < str.Length && flag!=true; i++)
        {
            if (str[i].ToString() == value)
            {
                head = head.next;
                  index = ++i;
                FinalState(str, head.data);
            }
else              {
                flag = true;
            }
        }
    }
    public static void FinalState(string str, string value)
    {
        Console.WriteLine("The word is in the language");
        flag = true;
    }
  }
}
```

03 “C# code for NFA to DFA Conversion”

04 “C# code of Finite Automata for Pattern Searching”

```
namespace PatternSearching
{    static class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Finite Automata for Pattern Searching\n");                    Console.WriteLine("Enter
a String");
string text = Console.ReadLine();
Console.WriteLine("Enter a Pattern");
string pattern = Console.ReadLine();
List<int> indexes = Search(text, pattern);
foreach (var item in indexes)
         {
            Console.WriteLine("Pattern found at index {0}",item);
         }
      }
      public static List<int> Search(string text,string pattern)
      {
         if (String.IsNullOrEmpty(pattern))
            throw new ArgumentException("the string to find may not be empty", "value");
         List<int> indexes = new List<int>();
for (int index = 0; ; index += pattern.Length)
         {
            index = text.IndexOf(pattern, index);
if (index == -1)
return indexes;
indexes.Add(index);
         }
      }
   }
}
```

---

05 “C# code to check if a given String is Palindrome”

```
using System;
namespace palindrome
{
   class Program
   {
      static void Main(string[] args)
      {
         string s,revs="";
         Console.WriteLine(" Enter string");
```

```
        s = Console.ReadLine();
        for (int i = s.Length-1; i >=0; i--) //String Reverse
        {
           revs += s[i].ToString();
        }
        if (revs == s) // Checking whether string is palindrome or not
        {
           Console.WriteLine("String is Palindrome \n Entered String Was {0} and reverse string is {1}", s, revs);
        }
        else
        {
           Console.WriteLine("String is not Palindrome \n Entered String Was {0} and reverse string is {1}", s, revs);
        }
        Console.ReadKey();
     }
  }
```