

K Lavanya Sai Kumar
klavanyasaikumar@gmail.com

8019616883

Github : <https://github.com/luvsai/nityoInfoTech/tree/main>

Software Backend Development Internship (.Net core 5.0)- Assignment

ANSWERS :

Part 1:

Title: Technical screening

Technology: Data Structures

1. Write a program which reads an integer value for seconds and converts it to h:m:s format, where h, m, s denote hours, minutes (less than 60) and seconds (less than 60) respectively

Input	An integer S is given in a line
Output	Print h, m and s separated by ':'. You do not need to put '0' for a value, which consists of a digit.
Constraints	$0 \leq S \leq 86400$
Sample input 1	46979
Sample output 1	0.543738426

C# code:

```
//Author K lavanya Sai kumar

using System;

class Time
{
    // function to find the power of number n
    static void displaytime(int seconds)
    {
        //get hours
        int hrs = (int)(Math.Floor((float)(seconds / 3600)));

        //get minutes
        seconds = (int)seconds % 3600;
        int mins = (int)(Math.Floor((float)(seconds / 60)));

        //get remaining seconds

        int secs = (int) seconds % 60;

        //format the display time
    }
}
```

```
        string output = String.Format("{0}:{1}:{2}", hrs, mins, secs);

        Console.WriteLine(output);

    }

    //Main program
    public static void Main()
    {

        int seconds = Convert.ToInt32(Console.ReadLine());

        displaytime(seconds);

    }
}
```

Output :

```
E:\nityotech\programs>.\Time.exe
121
0:2:1

E:\nityotech\programs>
```

2. Given a number, check if it is a power of 2.

Input

First line of *input* contains T - number of test cases. It's followed by T lines, each line containing a single positive integer.

Constraints

$1 \leq T \leq 10000$

$1 \leq N \leq 1018$

Output

For each test case, print "True" or "False", separated by a new line.

Sample Input

5 -(T-number of test cases)

1
8
10
25
512

Sample Output

True
True
False
False
True

C# Code:

```
// Author K lavanya sai kumar
using System;

class Power2
{
    // function to find the power of number n
    static bool isPowerOfTwo(int n)
    {
        if (n == 0)
            return false;

        while (n != 1)
        {
            if (n % 2 != 0)
                return false;

            n = n / 2;
        }
        return true;
    }

    //Main program
    public static void Main()
    {
        string T = Console.ReadLine();
        int ntest = Convert.ToInt32(T);
        //Looping through test cases
        for (int i = 0; i < ntest; i++) {
            string tc = Console.ReadLine();

            int testcase = Convert.ToInt32(tc);

            Console.WriteLine(isPowerOfTwo(testcase) ? "True" : "False");
        }
    }
}
```

```
}  
}
```

Output :

```
E:\nityotech\programs>.\Power2.exe  
2  
64  
True  
39  
False  
  
E:\nityotech\programs>
```

3. Which of the following properties does a simple graph not hold?

- A. Must be connected
- B. Must be unweighted
- C. Must have no loops or multiple edges
- D. Must have no multiple edges

Answer : Option A

4. A connected planar graph having 6 vertices, 7 edges contains _____ regions.

- A. 15
- B. 3
- C. 1
- D. 11

Answer : Option B

5. What are the main differences between an Array and a Dictionary?

Array	Dictionary
1. It is collection of elements of same data Type	1. It is collection of key value pairs
2. Every element is accessed using index	2. Elements are accessed by specifying the key Each key is unique in nature
3. array is contiguous and continuous memory locations are assigned in general	3. Dictionary is dynamic in nature; it grows in size as the program grows.
4 Arrays are Immutable	4. Dictionaries are Mutable.

5 used to implement many data structures

5. Used to search data using some other data as keys (Mapping) .

Part 2: (Optional)

Title: Technical screening

Technology: ASP.NET Core 5.0 - Web API

Description: Prepare a web API with following details.

1. Add a model class (use the below employee class)

```
public class Employee
{
    public int EmployeeId {get;set}
    public string EmployeeName {get;set}
    public string Address {get;set}
    public string Department {get;set}
}
```

2. Add a controller

3. Methods to be implemented are:

- a. GetAllEmployees
- b. GetEmployeeDetails

4. Route URL

- a. /api/employees
- b. /api/{id}

5. Initialize employee details with following data:

```
[
    {EmployeeId = 1, EmployeeName = "Mukesh Kumar", Address = "New Delhi", Department = "IT"},
    {EmployeeId = 2, EmployeeName = "Banky Chamber", Address = "London", Department = "HR"},
    {EmployeeId = 3, EmployeeName = "Rahul Rathor", Address = "Laxmi Nagar", Department = "IT"},
    {EmployeeId = 4, EmployeeName = "YaduVeer Singh", Address = "Goa", Department = "Sales"},
    {EmployeeId = 5, EmployeeName = "Manish Sharma", Address = "New Delhi", Department = "HR"}
]
```

6. Testing

Call the web API with Postman.

<http://localhost:port/api/employee>. Should return list of employee initialized

<http://localhost:port/api/employee/4> should return the employee with id=4

1. Model class : api.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Office.Models
{
    public class api
    {
        public int EmployeeId { get; set; }
        public string EmployeeName { get; set; }
        public string Address { get; set; }
        public string Department { get; set; }
    }
}
```

```
}  
}
```

2. Controller : apiController.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.Mvc;  
using Office.Models;  
using System.Collections;  
namespace Office.Controllers  
{  
    public class apiController : Controller  
    {  
        public api[] Employees;  
  
        //Constructor for initializing the employee details  
        public apiController()  
        {  
            //Initializing the user details  
            Employees = new api[5];  
            Employees[0] = new api() { EmployeeId = 1, EmployeeName = "Mukesh Kumar", Address =  
"New Delhi", Department = "IT" };  
            Employees[1] = new api() { EmployeeId = 2, EmployeeName = "Banky Chamber", Address  
= "London", Department = "HR" };  
            Employees[2] = new api() { EmployeeId = 3, EmployeeName = "Rahul Rathor", Address =  
"Laxmi Nagar", Department = "IT" };  
            Employees[3] = new api() { EmployeeId = 4, EmployeeName = "YaduVeer Singh", Address  
= "Goa", Department = "Sales" };  
            Employees[4] = new api() { EmployeeId = 5, EmployeeName = "Manish Sharma", Address  
= "New Delhi", Department = "HR" };  
  
        }  
  
        //function GetAllEmployee  
        public ArrayList GetAllEmployees()  
        {  
  
            int listlen = Employees.Length;  
  
            ArrayList employeeList = new ArrayList();  
            int index;  
  
            for ( index = 0; index < listlen; index++) {  
                employeeList.Add(  
                    new  
                    {  
                        EmployeeId = Employees[index].EmployeeId,  
                        EmployeeName = Employees[index].EmployeeName,  
                        Address = Employees[index].Address,  
                        Department = Employees[index].Department  
                    });  
            }  
  
            return employeeList;  
        }  
  
        public ArrayList GetEmployeeDetails(int Employeeid)  
        {  
            int listlen = Employees.Length;
```

```

        ArrayList employeeDetails = new ArrayList();
        if (Employeeid < listlen && Employeeid >= 0)
        {

            employeeDetails.Add(new
            {
                EmployeeId = Employees[Employeeid].EmployeeId,
                EmployeeName = Employees[Employeeid].EmployeeName,
                Address = Employees[Employeeid].Address,
                Department = Employees[Employeeid].Department
            });

        }
        return employeeDetails;
    }

    // GET: api/employee http://localhost:port/api/employee.
    public ActionResult employee(int id=-1)
    {
        // GET: api/employee http://localhost:port/api/employee.
        if (id == -1)
        {
            ArrayList employeeList = GetAllEmployees();
            return Json( employeeList, JsonRequestBehavior.AllowGet);
        }

        // GET: api/employee/4 http://localhost:port/api/employee/4.
        try
        {

            ArrayList employeeDetails = GetEmployeeDetails(id - 1);
            return Json(employeeDetails, JsonRequestBehavior.AllowGet);

        }
        catch (Exception e) {
            return Content("Request can't be Served");
        }

    }

}

```

6. Testing

Call the web API with Postman.

<http://localhost:port/api/employee>. Should return list of employee initialized

<http://localhost:port/api/employee/4> should return the employee with id=4

Search Postman

GET https://localhost:44313/api/employee

No Environment

Save

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results

Status: 200 OK Time: 58 ms Size: 763 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "EmployeeId": 1,
3   "EmployeeName": "Mukesh Kumar",
4   "Address": "New Delhi",
5   "Department": "IT"
6 },
7 {
8   "EmployeeId": 2,
9   "EmployeeName": "Bhaskar Chandra",
10  "Address": "London",
11  "Department": "HR"
12 },
13 {
14  "EmployeeId": 3,
15  "EmployeeName": "Rahul Rathor",
16  "Address": "Laxmi Nagar",
17  "Department": "IT"
18 },
19 },
20 }
```

Search Postman

GET https://localhost:44313/api/employee/4

No Environment

Save

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results

Status: 200 OK Time: 7 ms Size: 417 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "EmployeeId": 4,
3   "EmployeeName": "YaduVeer Singh",
4   "Address": "Goa",
5   "Department": "Sales"
6 }
7
8 }
```

-----End of Assignment -----