**DAY 13 ASSIGNMENT PRESENTED**

**BY**

**POTUKANUMA JEEVITHA**

**09-02-2022**

|  |
| --- |
| **1.Declare a 2 dimentional array of size (2,2) and initialize using indexes and print the values using nested for loop.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_2\_dimentional\_array\_2\_2\_using\_nestedfor\_loop  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[2, 2];  data[0, 0] = 7;  data[0, 1] = 14;  data[1, 0] = 8;  data[1, 1] = 30;  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write(data [i, j] + " ");  }  Console.Write(" \n ");  Console.ReadLine();  }  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **2.Declare a 2-D Array of size (3,2) and initialize in the same line while declaring and print the values using nested for loop.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_2\_dimentional\_array\_3\_2\_using\_nestedfor\_loop  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[,] { { 7, 8 }, { 14, 11 }, { 11, 30 } };  //data[0, 0] = 7;  //data[0, 1] = 8;  //data[1, 0] = 14;  //data[1, 1] = 11;  //data[2, 0] = 11;  //data[2, 1] = 30;  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write(data[i, j] + " ");  }  Console.Write(" \n ");  Console.ReadLine();  }  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **3.Declare a 2-D Array of size (3,3) and print trace of the array.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_2D\_ARRAY\_3\_3\_and\_print\_trace  {  internal class Program  {  static void Main(string[] args)  {  int sum = 0;  int[,] data = new int[3, 3];  data[0, 0] = 7;  data[0, 1] = 8;  data[0, 2] = 3;  data[1, 0] = 14;  data[1, 1] = 11;  data[1, 2] = 7;  data[2, 0] = 11;  data[2, 1] = 30;  data[2, 2] = 8;  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  if (i == j)  sum = sum + data[i, j];  }      }  Console.WriteLine(sum);  Console.ReadLine();  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **4. Declare a 2-D Array of size (2,2) and read values from user and print the array values.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_2D\_array\_2\_2\_user\_values\_and\_print  {  internal class Program  {  static void Main(string[] args)  {  int[,] data = new int[2, 2];  //read  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.WriteLine($"enter any value at ({i},{j})");  data[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  //print  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write(data [i, j] + " ");  }  Console.WriteLine();  }    Console.ReadLine();  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **5. Declare two 2-D array of size (2,2) and read values from user and print the sum of the two matrices.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace two\_2D\_array\_\_2\_2\_sum\_of\_2\_matrices  {  internal class Program  {  static void Main(string[] args)  {  int[,] a1 = new int[2, 2];  Console.WriteLine("enter first matrix");  //read  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  a1[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  int[,] a2 = new int[2, 2];  Console.WriteLine("enter second matrix");  //read  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  a2[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  Console.WriteLine("addition of two matrices");  int[,]a3 = new int[2, 2];  for (int i = 0; i < 2; i++)  for (int j = 0; j < 2; j++)  a3[i, j] = a1[i, j] + a2[i, j];  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write(a3[i, j] + " ");  }  Console.Write("\n\n");  }  Console.ReadLine();  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **6. Declare two 2-D array of size (2,2) and read values from user and print the product of the two matrices.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace two\_2D\_array\_2\_2\_product\_of\_2\_matrices  {  internal class Program  {  static void Main(string[] args)  {  int[,] a1 = new int[2, 2];  Console.WriteLine("enter first matrix");  //read  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  a1[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  int[,] a2 = new int[2, 2];  Console.WriteLine("enter second matrix");  //read  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  a2[i, j] = Convert.ToInt32(Console.ReadLine());  }  }  //product of 2 matrices  Console.WriteLine("product of two matrices");  int[,] a3 = new int[2, 2];  for (int i = 0; i < 2; i++)  for (int j = 0; j < 2; j++)  a3[i, j] = a1[i, j] \* a2[i, j];  for (int i = 0; i < 2; i++)  {  for (int j = 0; j < 2; j++)  {  Console.Write(a3[i, j] + " ");  }  Console.Write("\n\n");  }  Console.ReadLine();  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **7. what is a jagged array? What is the benefits of jagged array.** |
| **Ans:-** |
| **Jagged array:**  A jagged array is an array whose elements are arrays, possibly of different sizes. A jagged array is sometimes called an array of arrays. |
| **Benefits of jagged array:**   * It makes things easy where there is a need to store data in a multidimensional way using the same variable name. * It helps in memory management which makes the program to be executed very smoothly and fast as well. |

|  |
| --- |
| **8. WACP to declare a jagged array and print values.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace jagged\_array\_print\_values  {  internal class Program  {  static void Main(string[] args)  {  int[][] values = new int[3][];  values[0] = new int[] { 7, 8, 6 };  values[1] = new int[] { 10, 12, 14, 11, 23 };  values[2] = new int[] { 14, 11, 30, 3, 7, 8, 78 };  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < values [i].Length; j++)  {  Console.Write (values [i][j]+" ");  }  Console.Write("\n");    }  Console.ReadLine();  }  }  } |
| **Output:-** |
|  |

|  |
| --- |
| **9. what is Recursion?** |
| **Ans:-** |
| **Recursion:**  A function called itself repeatedly until a specified condition is satisfied. |

|  |
| --- |
| **10. WACP to illustrate usage of Recursion . what are the benefits of recursion.** |
| **Code:-** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace factorial\_using\_recursion  {  internal class Program  {  static int Factorial(int n)  {  int fact = 1;  for (int i = 1; i <= n; i++)  fact = fact\* i;  return fact;  }  static void Main(string[] args)  {  Console.WriteLine("enter any number");  int input= Convert.ToInt32(Console.ReadLine());  int fact = Factorial(input);  Console.WriteLine("{0} factorial {1}", input, fact);      Console.ReadLine();  }  }  } |
| **Output:-** |
|  |
| **Benefits:**   * Recursion adds clarity and reduces the time needed to write and debug code. * Reduces time complexity. * Performs better in solving problems based on tree structure. |

|  |
| --- |
| **11. WACP to illustrate usage of Stack<> write couple of points about stack.** |
| **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace stack\_\_using\_pop  {  internal class Program  {  static void Main(string[] args)  {  Stack<int> data = new Stack<int>();  data.Push(7);  data.Push(8);  data.Push(9);  Console.WriteLine(data.Count);  Console.WriteLine(data.Pop());  Console.WriteLine(data.Count);  Console.ReadLine();  }  }  } |
| **OUTPUT:** |
| **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace stack\_using\_peek  {  internal class Program  {  static void Main(string[] args)  {  Stack<int> data = new Stack<int>();  data.Push(7);  data.Push(8);  data.Push(9);  Console.WriteLine(data.Count);  Console.WriteLine(data.Peek());  Console.WriteLine(data.Count);  Console.ReadLine();  }  }  } |
| **OUTPUT:** |
| **ABOUT STACK:**   * Stack represents LAST IN FIRST OUT. * When you add an item in the list , it is called pushing the element. * When you remove it, it is called popping the element. * In stack, you are allowed to store duplicate elements. * A stack accept null as a valid value for reference types. |

|  |
| --- |
| **12. WACP to illustrate usage of Queue<> write couple of points about Queue.** |
| **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace queue\_using\_dequeue  {  internal class Program  {  static void Main(string[] args)  {  Queue<int> data = new Queue<int>();  data.Enqueue(17);  data.Enqueue(28);  data.Enqueue(39);  Console.WriteLine(data.Count);  Console.WriteLine(data.Dequeue());  Console.WriteLine(data.Count);  Console.ReadLine();    }  }  } |
| **OUTPUT:** |
| **CODE:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace QUEUE\_USING\_PEEK  {  internal class Program  {  static void Main(string[] args)  {  Queue<int> data = new Queue<int>();  data.Enqueue(17);  data.Enqueue(28);  data.Enqueue(39);  Console.WriteLine(data.Count);  Console.WriteLine(data.Peek());  Console.WriteLine(data.Count);  Console.ReadLine();  }  }  } |
| **OUTPUT:** |
| **About queue:**   * Queue represents FIRST IN FIRST OUT. * When you add an item in the list, it is called enqueue. * When you remove an item in the list, it is called dequeue. * As element are added to a queue , the capacity is automatically increased as required by reallocating the internal array. * Queue accepts null as a valid value for reference types and allows duplicate elements. |