**דף עזר לבחינה**

|  |  |  |
| --- | --- | --- |
| הגדרת משתנה |  | int grade |
| משפט if |  | if (grade > 60) {  Console.WriteLine(“Pass”)  }  else  {  Console.WriteLine(“Fail”)  } |
| משפט switch |  | switch (a)  {  case 0:  Console.WriteLine("zero");  break;  case 1:  Console.WriteLine("one");  break;  default:  Console.WriteLine("default");  break;  } |
| string s = “programming” | 🡪 true | s.EndsWith(“ing”) |
|  | 🡪 true | s.StartsWith(“pr”) |
|  | 🡪 10 | s.IndexOf(“g”, 6) |
|  | 🡪 “the programming” | s.Insert(0, “the “) |
|  | 🡪 “pgraming” | s.Remove(1, 2) |
|  | 🡪 “program” | s.Replace(“ing”, ““) |
|  | 🡪 “ram” | s.Substring(4, 3) |
| int i = 5, double d = 7.7,  string s = “2” | 🡪 i = 7 | i = Convert.ToInt32(d) |
|  | 🡪 i = 2 | i = Int32.Parse(s) |
|  | 🡪 s = “5” | s = i.ToString() |
| לולאת for |  | for (int i = 1; i < 10; i++)  {  Console.WriteLine(i);  } |
| לולאת while |  | while(a < 4)  {  a = a + 1;  } |
| הגדרת מערך |  | int[,] integerArray = new int[10,5];  integerArray.Length;  integerArray.GetLength(0);  integerArray.GetLength(1); |
| Special characters |  | -\n: enter a new line -\t: insert a tab -\b: one char back -\\: print the char \ -\": print the char " |
| Converting inputs |  | int.Parse (Console.ReadLine()) double.Parse (Console.ReadLine()) |
| casting |  | (int)  (double) |
| Copy arrays | \*Src = the source array  \*Dest = the destination array \*length = amount of elements to copy | .CopyTo(array, index)  .Copy(src, dest, length)  .Clone() – we need to cast the return type. |
| Sorting an array | It sorts the elements in the array according to their type and order | Array.Sort() |
| Array clear | Index – where to start  Length – how many elements to clear | Clear(array,index,length) |
| foreach |  | Foreach (type varName in ArrName) {  Use varName here as the current value  } |
| rank | will return the number of dimensions –in our | [array name].Rank |
| clone | will make and return another copy of the matrix | (casting)[array name].Clone() |
| Stack | Push – add an element  Pop – remove an element | LIFO – last in, first out |
| Queue | Enqueue – add an element Dequeue - remove an element | FIFO – first in last out |