**Question No.1: Write a program that takes 10 words (strings) from user and sorts elements in lexicographical (dictionary) order. The program will contain a function for sorting (function name sorting)**

**INPUT:**

public static void Main()

{

Console.WriteLine("Without Dictionary Sorting :\n");

string[] array = new string[10]{"Zeeshan","Ali","Abbas","Kamran","Merzia","Farhat","Shahid","Nosheen","Shaheer","Shaheen"};

Console.WriteLine("Array before sorting");

for (int i = 0; i < array.Length; i++)

{

Console.WriteLine(array[i]);

}

Sorting(array, 0, (array.Length - 1));

Console.WriteLine("\nDictionary Sorting : \n");

for (int i = 0; i < array.Length; i++)

{

Console.WriteLine(array[i]);

}

}

public static void Sorting(string[] A, int LB, int UB)

{

if (LB <= UB)

{

int Pv = Partition(A, LB, UB);

Sorting(A, LB, Pv - 1);

Sorting(A, Pv + 1, UB);

}

return;

}

public static int Partition(string[] A, int LB, int UB)

{

int Pv = LB, Lo = LB + 1, Hi = UB;

while (Lo <= UB && Hi >= LB)

{

if (A[Lo].CompareTo(A[Pv]) < 0)

{

Lo++; continue;

}

if (A[Hi].CompareTo(A[Pv]) > 0)

{

Hi--; continue;

}

if (Lo < Hi)

{

string temp1 = A[Hi];

A[Hi] = A[Lo];

A[Lo] = temp1;

}

else

{

break;

}

}

string temp = A[Pv];

A[Pv] = A[Hi];

A[Hi] = temp;

Pv = Hi;

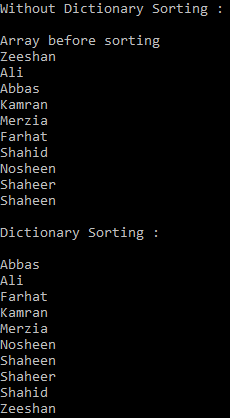
return Pv;

}

}

}

**OUTPUT:**



**Question 2: A positive integer is input, write a function (function name: binary) to find the binary equivalent of this number using recursion. For example, if input is 156, then binary value is 10011100 (no Java code is needed, just a sketch and pseudo code).**

**SKETCH:**

**First:**

|  |
| --- |
| **Num=12** |

|  |
| --- |
| **Method(Num)** |

|  |
| --- |
| **If Num==0**  **(FALSE)**  **Return 0** |

|  |
| --- |
| **N=Num%2 (12%2=0)**  **N=0** |

|  |
| --- |
| **k=** **decimal\_to\_binary\_converter(Num / 2)**  **k=6** |

**Second:**

|  |
| --- |
| **If Num==0**  **(FALSE)**  **Return 0** |

|  |
| --- |
| **Num=6** |

|  |
| --- |
| **Method(Num)** |

|  |
| --- |
| **N=Num%2(6%2=0)**  **N=0** |

|  |
| --- |
| **k=** **decimal\_to\_binary\_converter(Num / 2)**  **k=3** |

**Third:**

|  |
| --- |
| **N=Num%2(3%2=1)**  **N=1** |

|  |
| --- |
| **Num=3** |

|  |
| --- |
| **Method(Num)** |

|  |
| --- |
| **If Num==0**  **(FALSE)**  **Return 0** |

|  |
| --- |
| **k=** **decimal\_to\_binary\_converter(Num / 2)**  **k=0** |

**Fourth:**

|  |
| --- |
| **N=Num%2(1%2=1)**  **N=1** |

|  |
| --- |
| **Method(Num)** |

|  |
| --- |
| **If Num==0(FALSE)**  **Return 0** |

|  |
| --- |
| **Num=1** |

|  |
| --- |
| **k=** **decimal\_to\_binary\_converter(Num / 2)**  **k=0** |

**Fifth:**

|  |
| --- |
| **Method(Num)** |

|  |
| --- |
| **If Num==0(TRUE)**  **Return 0** |

|  |
| --- |
| **Num=0** |

**CODE:**

public static void decimal\_to\_binary\_converter(int num)

{

if (num >0)

{

int n = num % 2;

decimal\_to\_binary\_converter(num / 2);

Console.Write(n);

}

}

public static void Main()

{

Console.WriteLine("ENter number to find lcm : ");

int num = int.Parse(Console.ReadLine()) ;

decimal\_to\_binary\_converter(num);

Console.WriteLine();

}

**PSEUDO CODE:**

**Main():**

Input: “number to convert it into Binary”;

**Call Method():**

Now after calling the method the condition will be checked

if (the number is equal to zero):

return zero(print 0)

else:

num%2

decimal\_to\_binary\_converter(num / 2)

**Explanation:**

* take modulus of the number with 2 i.e num%2
* divide the number by 2
* For recursion call method again i.e. decimal\_to\_binary\_converter(num / 2)
* the process will repeat again and again until we get num = 0
* when num becomes 0 the condition wil become true (if num==0)

Return the answer print it.

**QUESTION 3: Show how to implement a queue using 2 stacks (no Java code is needed, just a sketch and pseudo code**

**Stack-1 :initially empty(top = -1) Stack-2 : initially is empty**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| **10** | **<-- top index = 9** |
| **9** |
| **8** |
| **7** |
| **6** |
| **5** |
| **4** |
| **3** |
| **2** |
| **1** |

**Stack-1 : push values(1-10) Stack-2 : initially is empty**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Stack-1: pop values and push into stack 2**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| **1** | **<-- top** |
| **2** |
| **3** |
| **4** |
| **5** |
| **6** |
| **7** |
| **8** |
| **9** |
| **10** |

**Stack-1: Stack-2:**

**Now our Queue is ready**

**In queue concept of first In first out is followed (FIFO)**

**Psedo code:**

Let’s take ten(10) elements from the user or hard coded values from (1-10)  
Now declare two stacks  
Stack Stack1, Stack2;  
 Push all the values into Stack Stack1  
for( i=1; i <=10;i++)  
{  
S1.push(i);  
}  
 Stack1 contains {10,9,8,7,6,5,4,3,2,1}

Top=10  
  
pop the values in the Stack1 and Push it to stack2 until stack1 does not becomes empty  
while(!S1.isEmpty())  
{  
S2.push(S1.pop());  
}  
 Stack2 contains {1,2,3,4,5,6,7,8,9,10} where the top is 1  
  
You can now pop elements from Stack2. First element will be the first to pop and so on...

**Question 4: We have implemented in class the algorithm for converting infix to postfix using stacks create a similar method for prefix conversion.**

Reverse the expression

Convert every ‘(’ as ‘)’ and every ‘)’as ‘(’

Now scan each character sequentially

If character == operand copy it to prefix string output

If the character if opening parenthesis push it into stack and push other coming characters

If the character is closing parenthesis then pop all the elements until the opening parenthesis comes.

If the coming character is an operator and it has greater precedence or equal precedence then the operator present in the top of the stack then push it.

If the coming character is an operator and it has lesser precedence than the operator present in the top of the stack, then pop all the characters or operators and show them in the output string and push the new operator.

Now again reverse the output expression because if we will not reverse it then it will be a postfix expression in which operands are placed before operators

**Question 5: Add the following methods to linked list Class discussed during lecture** :

**a. Method get(int n), which should return the element of index 𝑛 (indexing starts with 0). If the index is out of bounds, the exception IllegalArgumentException should be thrown.**

**CODE:**

public int getMethod(int index) {

int count = 0;

Node temp = start.next;

while (temp!=null)

{

if (count==index)

{

Console.WriteLine(temp.data);

return temp.data;

}

count++;

temp = temp.next;

}

throw new ArgumentException("Index out of bound");

}

**b. Method insertAt(Item x, int n), which should insert an element at index n into the list. If the index is out of bounds, the exception IllegalArgumentException should be thrown**

**CODE:**

public void insertElementOnIndex(int data,int index) {

Node n = new Node(data);

int count = 0;

Node temp = start.next;

if (!underFlow())

{

while (temp.next != null)

{

if (count == index - 1)

{

n.next = temp.next;

temp.next = n;

break;

}

else

{

++count;

temp = temp.next;

}

}

if (temp.next==null)

{

throw new ArgumentException("Index out of bound");

}

}

}

**c. Method removeAt(int n), which should remove an element at index n from the list. If the index is out of bounds, the exception IllegalArgumentException should be thrown**

**CODE:**

public void deleteElementFormIndex(int index)

{

if (start.next == null)

return;

Node temp = start.next;

if (index==0)

{

start.next = temp.next;

return;

}

for (int i = 0; temp != null && i < index - 1; i++)

temp = temp.next;

if (temp == null || temp.next == null)

{

throw new ArgumentException("Index Out oF bound");

}

Node next = temp.next.next;

temp.next = next;

}