**Bahria University**

**Software Engineering Department**



**Course: CSC 220 DATA STRUCTURE & ALGORITHMS**

**Term: Fall 2019, Class: BSE 3(B)**

**Assignment No:**

|  |  |
| --- | --- |
| **0** | **2** |

**Submitted By:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **Q** | **I** | **B** |  | **M** | **E** | **H** | **M** | **O** | **O** | **D** |

**Enrollment No.:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2 | - | 1 | 3 | 1 | 1 | 8 | 2 | - | 0 | 1 | 4 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Submission Date** | | | | | | | |  | **Date Submitted** | | | | | | | |
| **0** | **6** | **/** | **1** | **2** | **/** | **1** | **9** | **0** | **6** | **/** | **1** | **2** | **/** | **1** | **9** |

**Submitted To:**

**Engr. Bushra Fazal**

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Max Marks: \_\_\_\_\_\_\_\_\_\_\_ Marks Obtained: \_\_\_\_\_\_\_\_\_\_\_\_\_**

Bahria University,

Karachi Campus



Assignment NO.

02.

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 01. | 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). |
| 02. | 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). |
| 03. | Show how to implement a queue using 2 stacks (no Java code is needed, just a sketch and pseudo code) |
| 04. | We have implemented in class the algorithm for converting infix to postfix using stacks create a similar method for prefix conversion. |
| 05. | Add the following methods to linked list Class discussed during lecture   1. 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. 2. 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. 3. 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. |

Submitted On:

06-12-2019

(Date: DD/MM/YY)

Q 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 Sorting(string[] input,int lb,int ub)

{

//lb=0,ub=9

if (lb < ub)

{

int pv = lb;

int lo = lb + 1;

int hi = ub;

while ((lo <= hi) && (hi >= lb) && (lo <= ub))

{

if (input[pv].CompareTo(input[lo]) > 0)

{

lo++;

continue;

}

if (input[pv].CompareTo(input[hi]) < 0)

{

hi--;

continue;

}

else

{

string temp = input[lo];

input[lo] = input[hi];

input[hi] = temp;

}

}

string temp1 = input[pv];

input[pv] = input[hi];

input[hi] = temp1;

pv = hi;

Sorting(input, lb, pv - 1);

Sorting(input, pv + 1, ub);

}

}

static void Main(string[] args)

{

Console.WriteLine("Enter 10 words OR Strings");

string[] input = new string[10];

for (int i = 0; i < 10; i++)

{

Console.Write("{0} = ",i+1);

input[i] = Console.ReadLine();

} Console.WriteLine();

Console.WriteLine("After Sorting in Dictionary Order");

int lb = 0;

int ub = input.Length - 1;

Sorting(input,lb,ub);

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

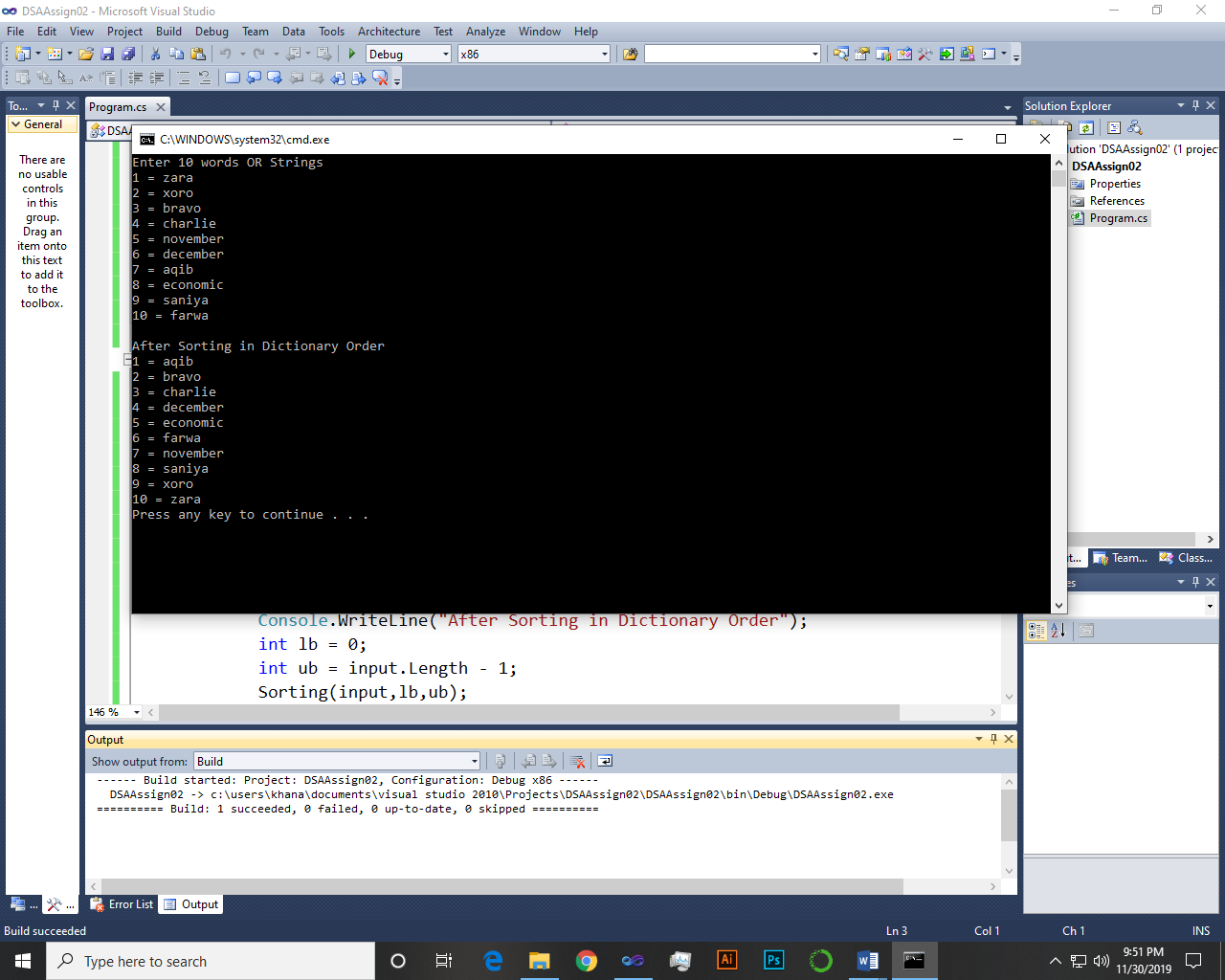
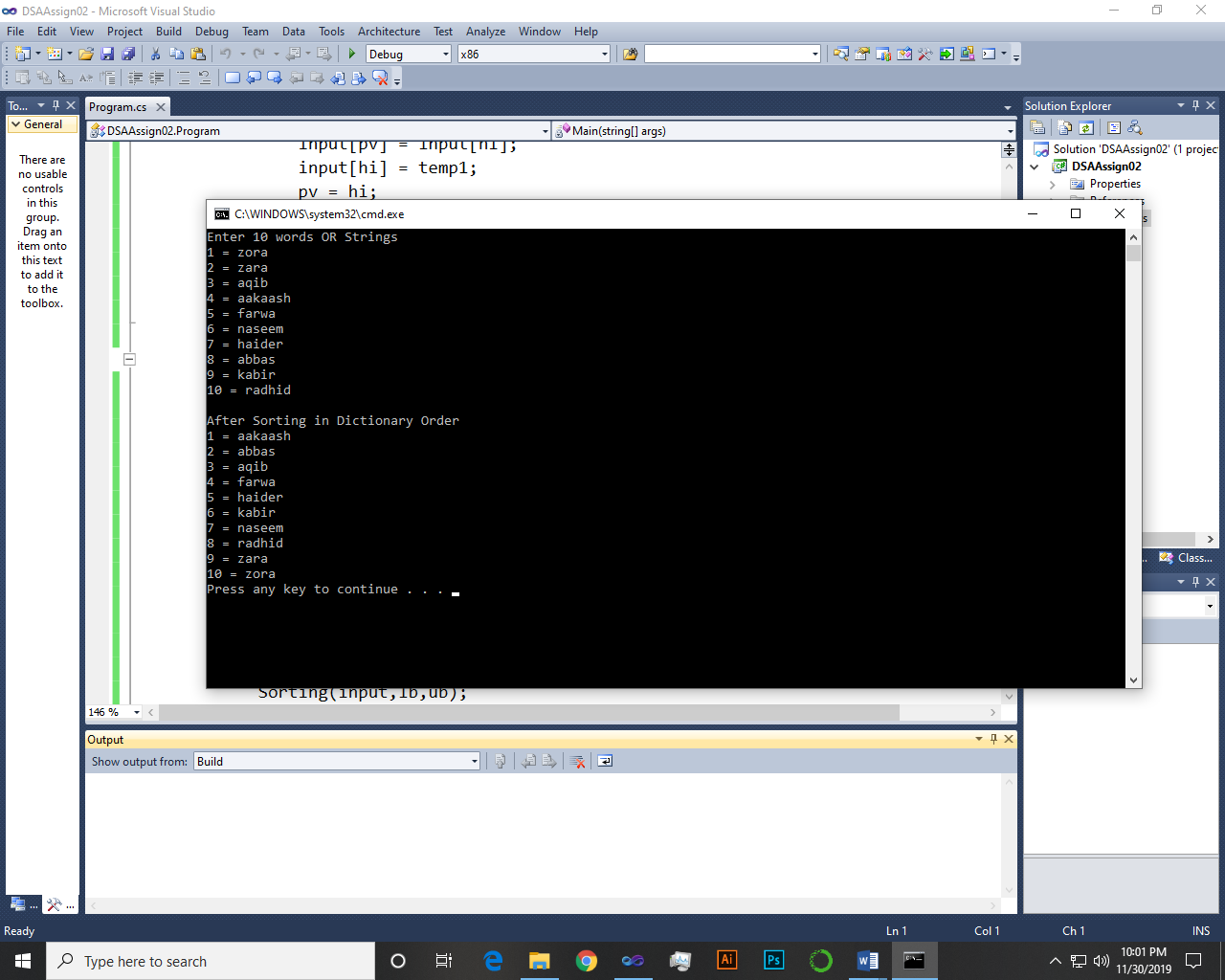
{

Console.WriteLine("{0} = {1}", i + 1,input[i]);

}

}

Output:-

~~~~~~\*\*/ /\*\*~~~~~~

Q 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).

Pseudo Code:-

\*Input a positive number "n" to find binary Equivalent

\* now call the function

\* Binary(n)

\* {

\* condition check

\* if(n>0)

\* return n % 2 + 10 \* Binary(n / 2)

\* recursive call to binary function

\* at the end this statement return the result in binary form

\* else return 0

\* }

\* So if the input number is 16 the Binary equivalent will be 10000

Sketch:-

|  |  |
| --- | --- |
| 2 | 16 |
| 2 | 8-0 |
| 2 | 4-0 |
| 2 | 2-0 |
| 2 | 1-0 |

C# Code:-

Console.WriteLine("Enter a positive number to find Binary Equivalent");

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

Console.WriteLine(Binary(n));

public static int Binary(int n)

{

if (n > 0)

{

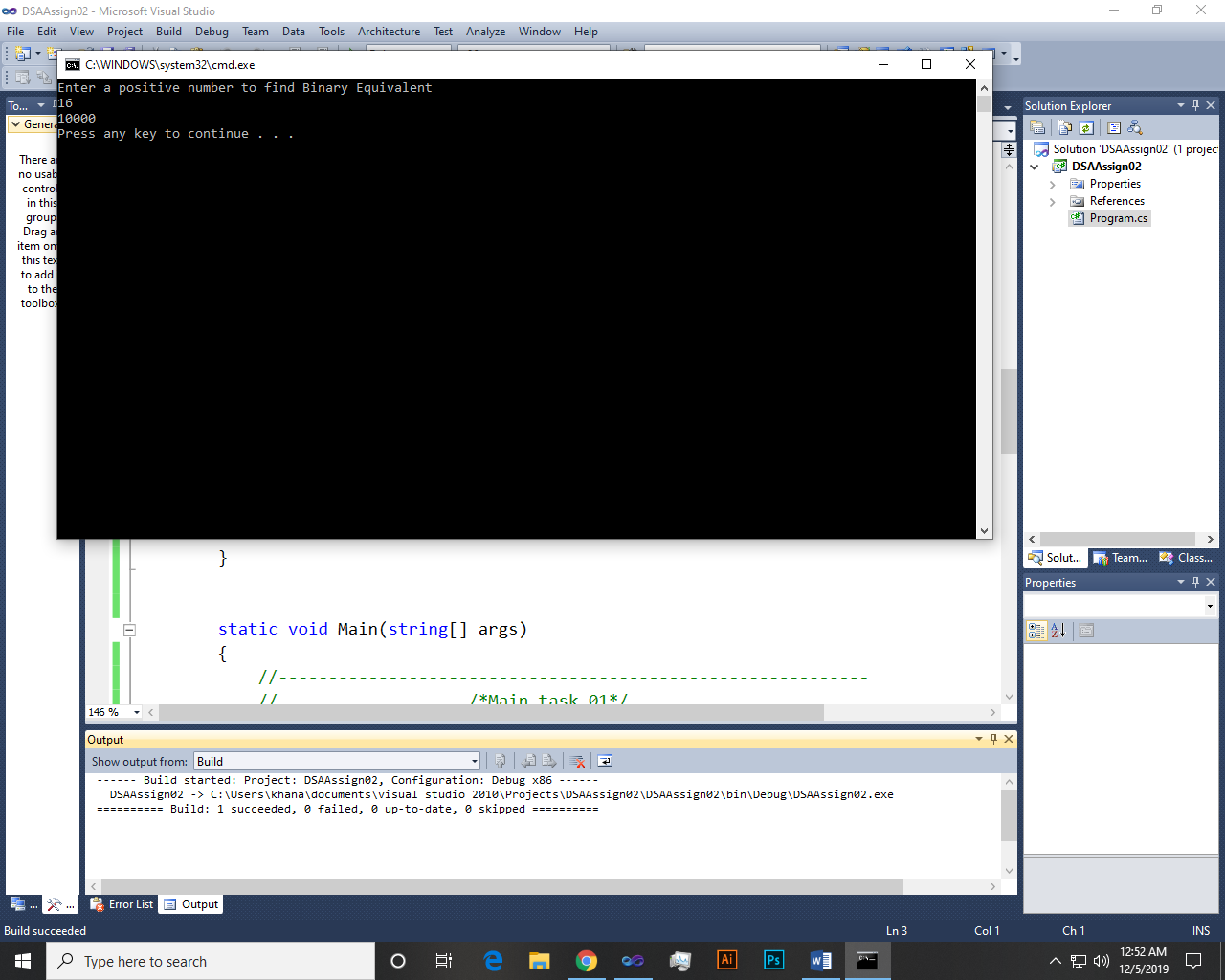
return n % 2 + 10 \* Binary(n / 2);

}

return 0;

}

Output:-



~~~~~~\*\*/ /\*\*~~~~~~

Q 3: - Show how to implement a queue using 2 stacks (no Java code is needed, just a sketch and pseudo code).

Input:-

* Queue is FIFO or LILO.
* We can easily implement a Queue using two Stacks.
* First of all create two stack classes namely Stack1 and Stack2.
* Here I use stack2 as a Queue.

**Enqueue:-**

enQueue(val)

* Push(enQueue) val into Stack2(Queue) from Stack1 if Stack1 is not empty.
* If Stack1 is empty then show error message.

**Dequeue:-**

deQueue()

* Pop(deQueue) a value from Stack2(Queue) if Stack2 is not empty.
* If Stack2 is empty then show error message.

Sketch:-

Enqueue

Push Pop

|  |
| --- |
|  |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

|  |
| --- |
|  |
| 5 |
| 4 |
| 3 |
| 2 |
| 1 |

|  |
| --- |
|  |
| 5 |
| 4 |
| 3 |
| 2 |
| 1 |

Stack1

Stack1

Stack2 as Queue

Dequeue

~~~~~~\*\*/ /\*\*~~~~~~

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

Algorithm for prefix conversion:-

*Note: - In prefix, operators should place before operands.*

1. In case of prefix conversion First of all Reverse the infix expression.
2. Now obtain the postfix expression of reversed infix expression.
3. Again reverse the postfix expression to obtain the final result that is Prefix Expression.
4. To obtain the postfix expression we must follow the below algorithm.

* Print operands as they arrive.
* If the stack is empty or contains an open parenthesis on top, push the incoming operator onto the stack.
* If the incoming symbol is an open parenthesis, push it on the stack.
* If the incoming symbol is a close parenthesis, pop the stack and print the operators until you see an open parenthesis. Discard the pair of parentheses.
* If the incoming symbol has higher precedence than the top of the stack, push it on the stack.
* If the incoming symbol has equal precedence with the top of the stack, use association. If the association is left to close, pop and print the top of the stack and then push the incoming operator. If the association is close to left, push the incoming operator.
* If the incoming symbol has lower precedence than the symbol on the top of the stack, pop the stack and print the top operator. Then test the incoming operator against the new top of stack.
* At the end of the expression, pop and print all operators on the stack. (No parentheses should remain.)

Sketch:-

Suppose we have an infix expression A\*(B + C \* D) +E now try to convert it into prefix expression using stack.

Step 1:-

Reverse the infix expression: E+(D\*C+B)\*A

Step 2:-

Now obtain the postfix expression of reversed infix expression.

| **S.No.** | **Expression** | **Stack** | **Prefix** |
| --- | --- | --- | --- |
| 1 | E |  | E |
| 2 | + | + | E |
| 3 | ( | +( | E |
| 4 | D | +( | E D |
| 5 | \* | +(\* | E D |
| 6 | C | +(\* | E D C |
| 7 | + | +(+ Pop\* | E D C \* |
| 8 | B | +(+ | E D C \* B |
| 9 | ) | + | E D C \* B+ |
| 10 | \* | +\* | E D C \* B+ |
| 11 | A | +\* | E D C \* B+A |
| 12 |  | Pop all elements | E D C \* B+A\*+ |

Step 3:-

Again reverse the postfix expression to obtain the final result that is Prefix Expression.

Prefix expression = +\*A+B\*CDE so, this is the final Prefix expression.

~~~~~~\*\*/ /\*\*~~~~~~

Q 5: - Add the following methods to linked list Class discussed during lecture

* 1. 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.
  2. 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.
  3. 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.

Solution:-

Get Method:-

public int Get(int n)

{

int counter = 0;

Node temp = start.next;

while (temp != null)

{

if (counter == n)

{

Console.WriteLine(temp.data);

return temp.data;

}

counter++;

temp = temp.next;

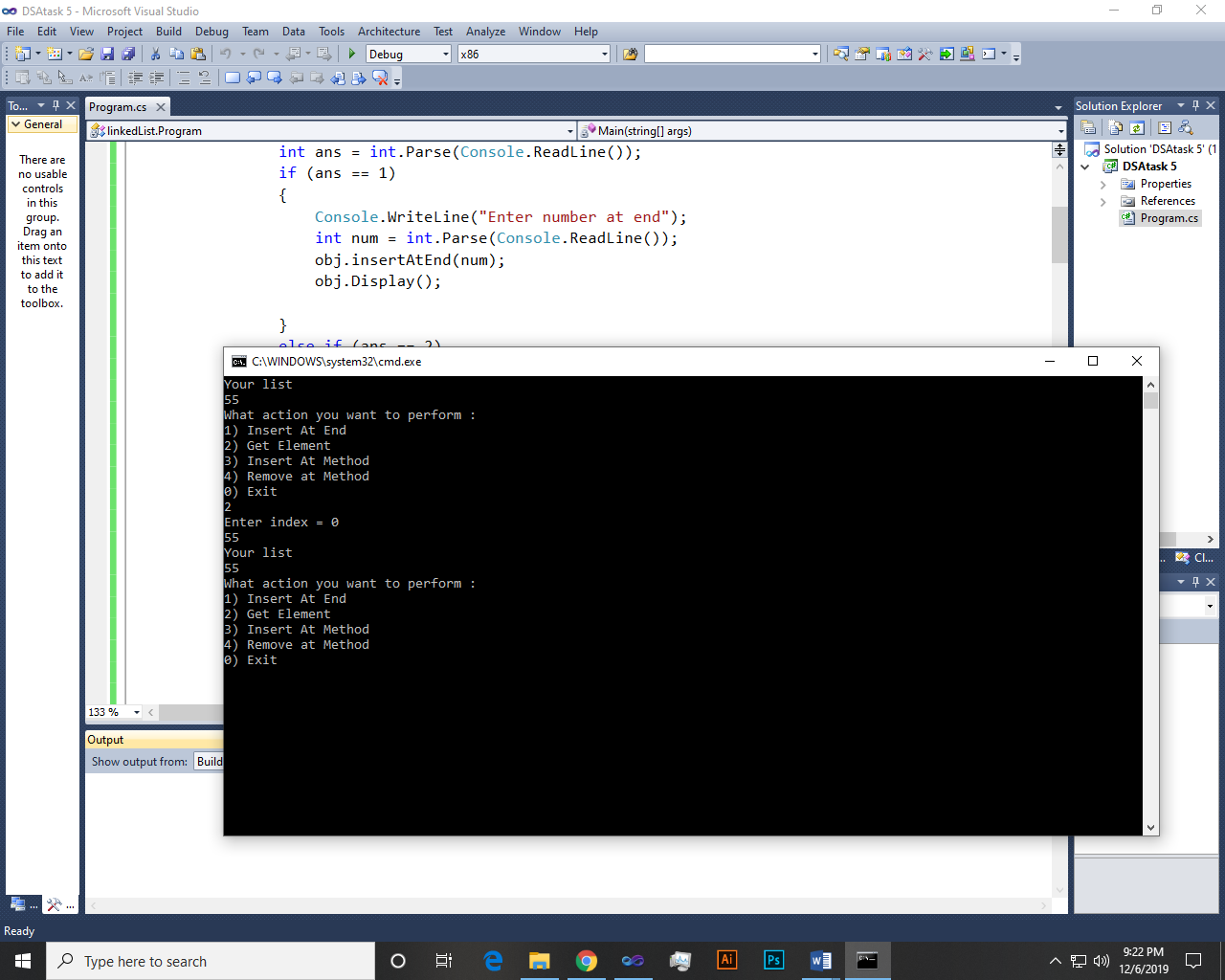
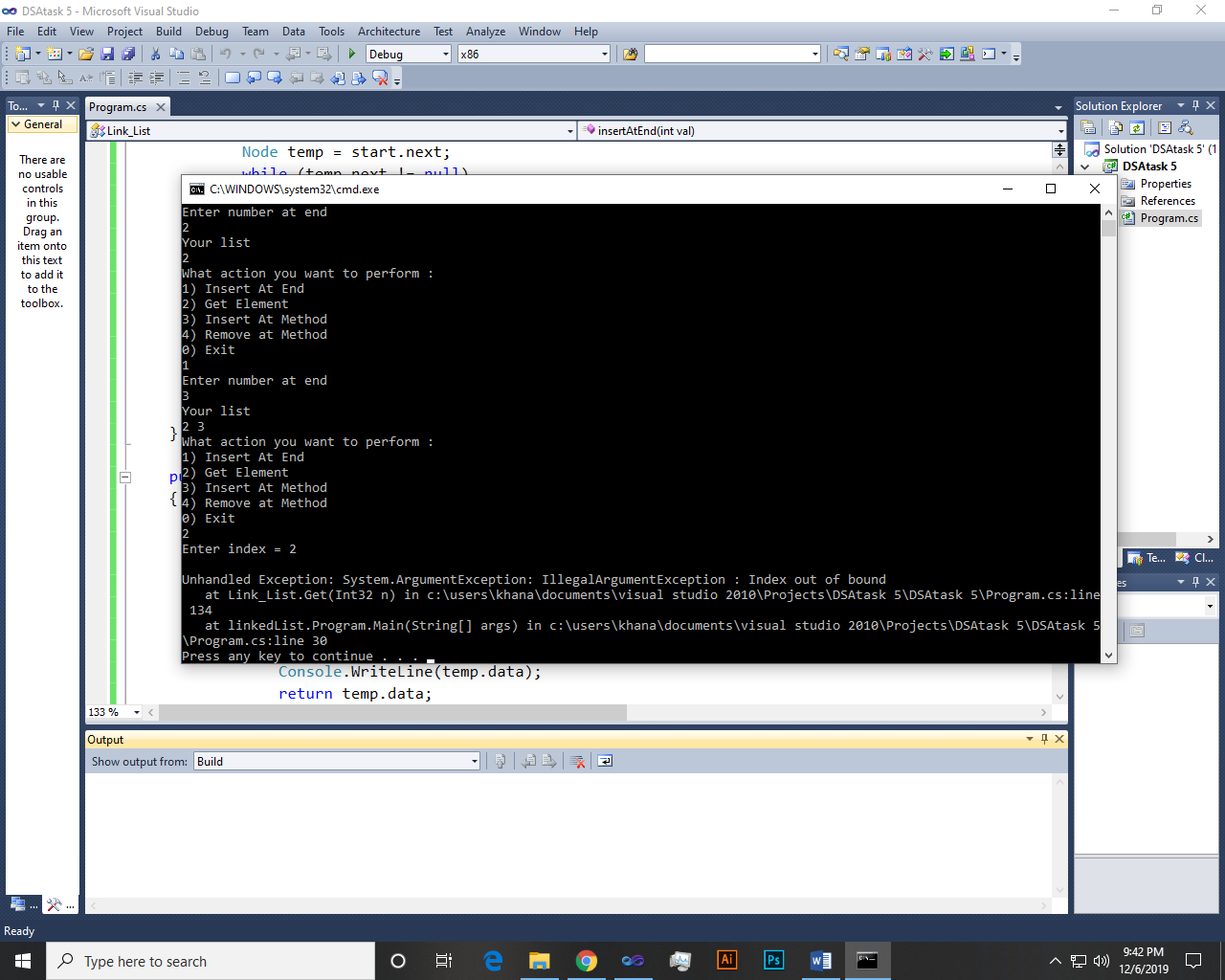
}

throw new ArgumentException("IllegalArgumentException :

Index out of bound");

}

Output:-

InsertAt Method:-

public void InsertAt(int x, int index)

{

Node n = new Node(x);

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("IllegalArgumentException :

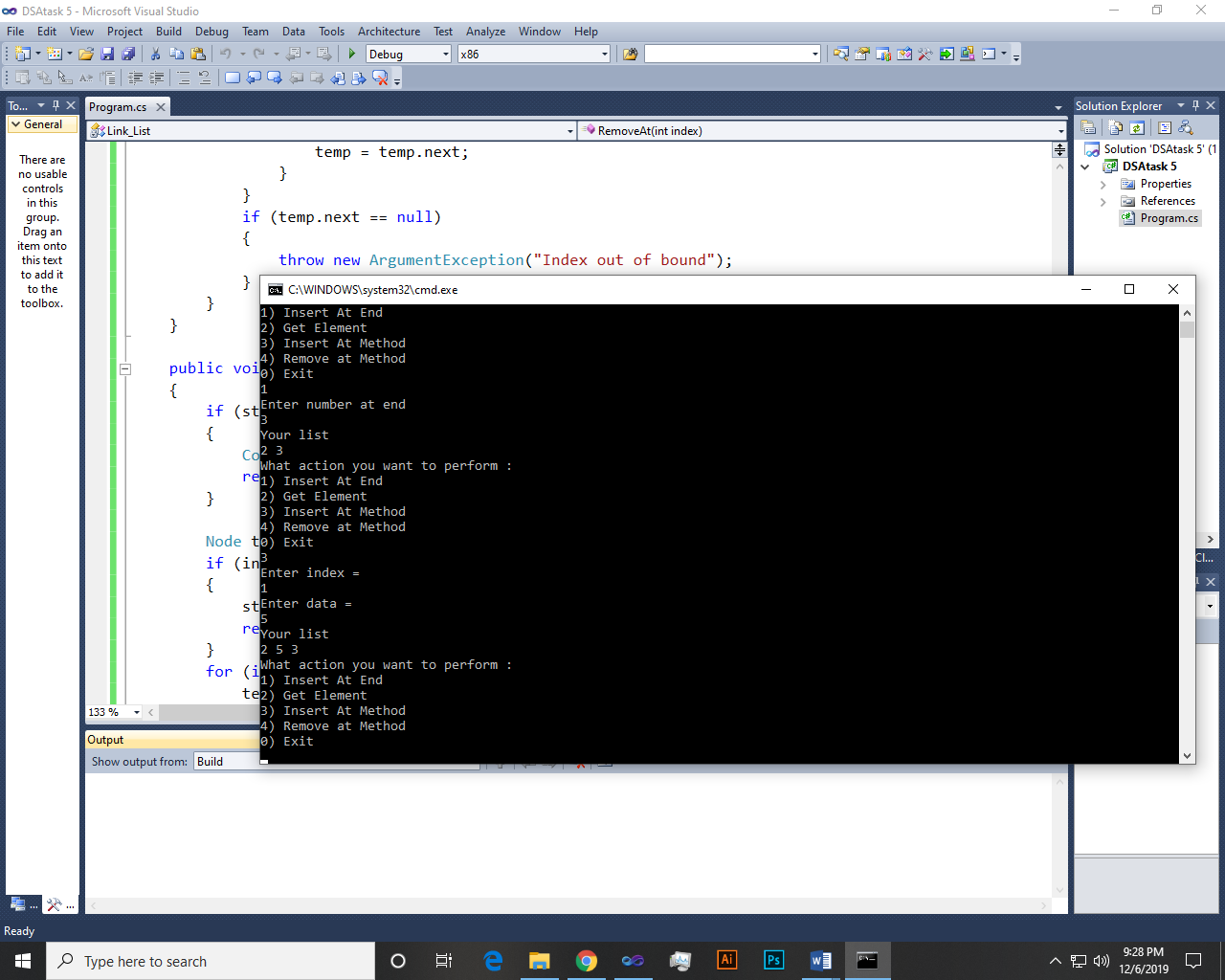
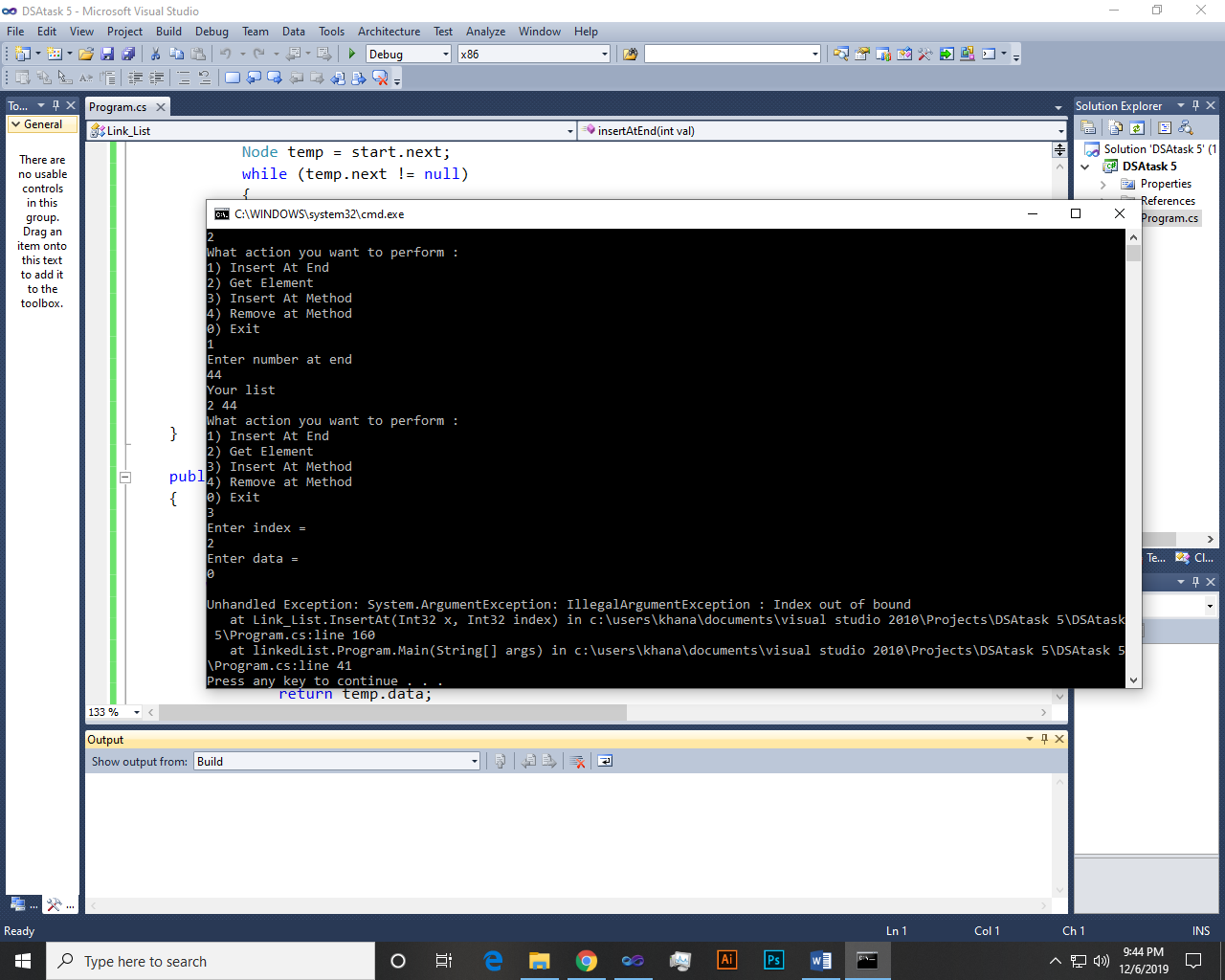
Index out of bound");

}

}

}

Output:-

RemoveAt Method:-

public void RemoveAt(int index)

{

if (start.next == null)

{

Console.WriteLine("List is Empty.");

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("IllegalArgumentException :

Index Out oF bound");

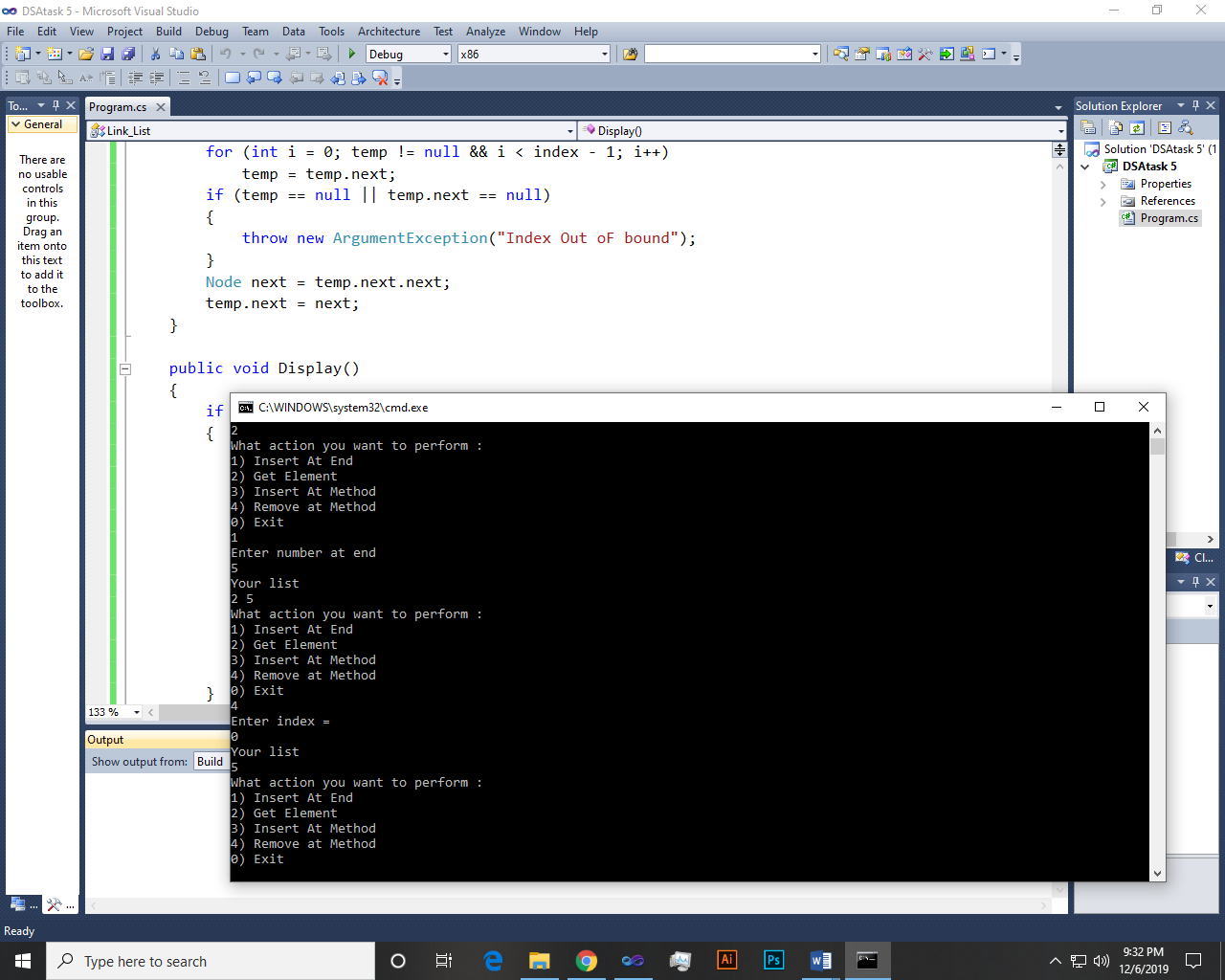
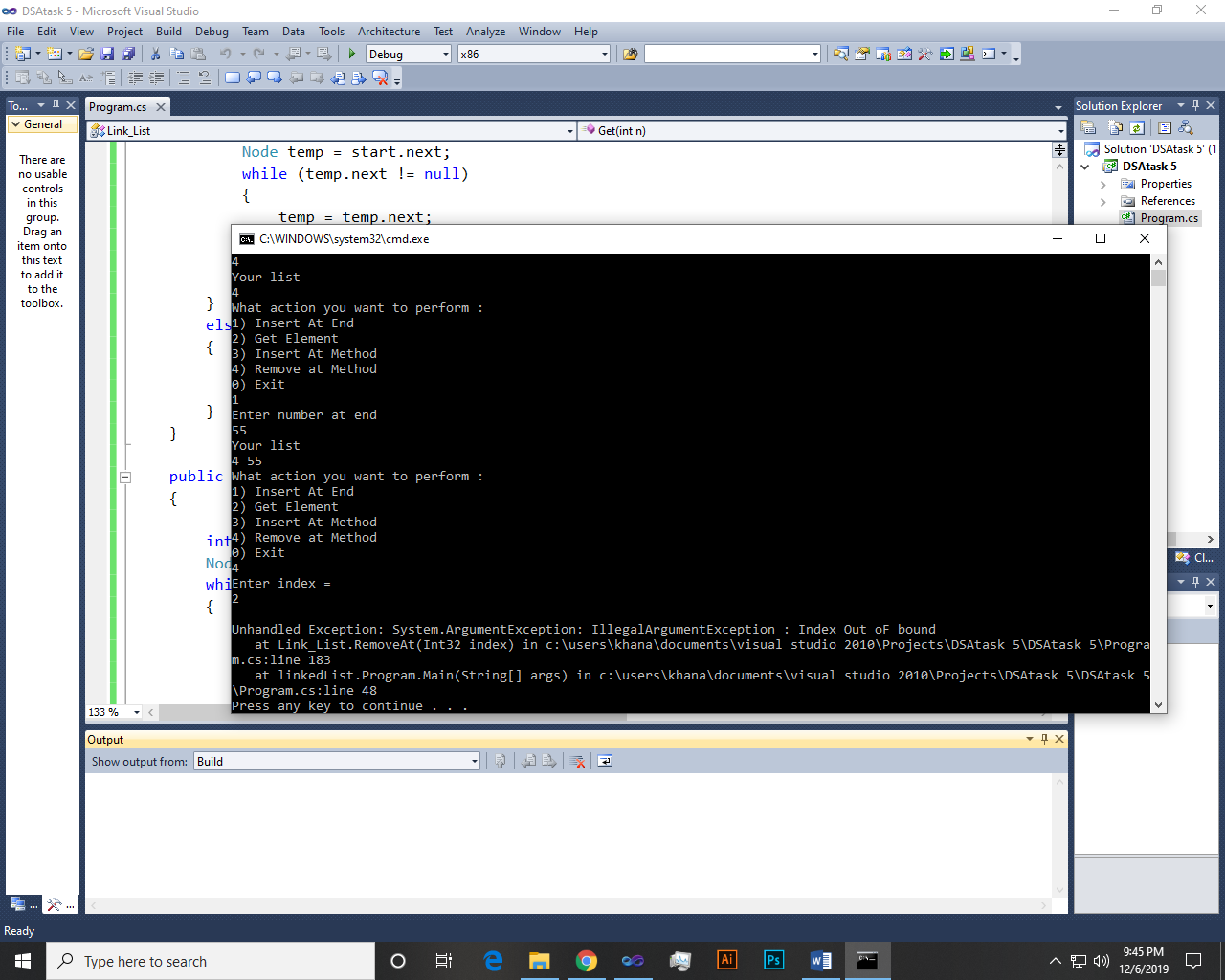
}

Node next = temp.next.next;

temp.next = next;

}

Output:-

~~~~~~\*\*/**THE END**/\*\*~~~~~~