***Project Assessment Documentation***

***On***

***Doubly Linked List***

***For the Assessment of***

**MCA First Semester (Shift-I)**

***In the Course***

**Concepts in Data Structures Lab (MCP531)**

***Developed by***

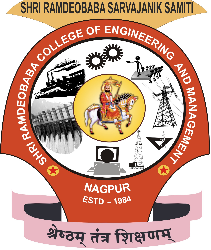
**Vaibhav Shewale**

**Akanksha Bhoyar**

***Under the Guidance of***

**Dr. Smita Nirkhi**

Assistant Professor**,** RCOEM



**Department of Computer Application**

**Shri Ramdeobaba College of Engineering & Management**

**Nagpur-13**

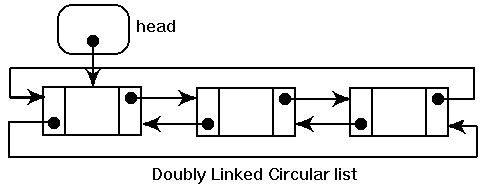
**Session: 2018-2019**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Particulars** | **Page No.** |
| **1.** | **Introduction** | **1** |
| **2.** | **Aim and objective** | **2** |
| **3.** | **Algorithms/ Mechanism Implemented** | **3** |
| **4.** | **Coding** | **6** |
| **5.** | **Input and Output Screens** | **12** |
| **6.** | **Conclusion** | **13** |

**Introduction**

A Doubly Linked List is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains three fields: two link fields (references to the previous and to the next node in the sequence of nodes) and one data field in between. The beginning and ending nodes' previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list. If there is only one sentinel node, then the list is circularly linked via the sentinel node. It can be conceptualized as two singly linked lists formed from the same data items, but in opposite sequential orders.



The two-node links allow traversal of the list in either direction. While adding or removing a node in a doubly linked list requires changing more links than the same operations on a singly linked list, the operations are simpler and potentially more efficient (for nodes other than first nodes) because there is no need to keep track of the previous node during traversal or no need to traverse the list to find the previous node, so that its link can be modified.

**Aim:** Application to split doubly linked list into 2 doubly linked list (all odd number will go to first list and all even numbers to second list).

**Objective:**

The objective of this program is to perform a doubly link list procedure by taking input from the user and saving it in doubly link list, but not only that but splitting that linked list to another doubly link list and performing a ‘odd-even’ function and showing the values in doubly linked list.

For this we have used Dev-C++ compiler as it’s a open source program and it has many new feature addons that helped us to perform our program.

**Algorithms**

Algorithm for structure node

Step 1: declare variable

Step 2: end

Global variable declaration

Algorithm for insert function to create a new node

Step 1: allocate memory for new node

Step 2: set the data field and link field of the new node

Step 3: create a new list if head pointer is empty

Step 4: insert the node at the end of the list

Step 5: end

Algorithm to display updated list in reverse direction

Step 1: print “last”

Step 2: display the list

Step 3: print “head”

Step 4: end

Algorithm for print updated list in forward direction

Step 1: print “head”

Step 2: display the list

Step 3: print “last”

Step 4:end

Algorithm for split list

Step 1: allocate memory for the new nodes

Step 2: Repeat step while(list != NULL)

2.1 set the data field and link field of node

2.2 if(listp->data%2 = 0)then

2.2.1: if(even=NULL) then

create a new list and add the node

else

add node in the even list

[end of if structure]

Else

if(odd=NULL) then

create a new list and add the node

else

add the node in the current odd list

[end of if structure]

[end of outer if structure]

[end of step 2 loop]

Step 3: if (listp!=NULL) then

allocate memory to link node

set the data and link field

if(listp->data%2 = 0) then

set current = even;

repeat while(current!=NULL)

current = current->next

[end loop]

Insert link at the end of list

Else

Current = odd

Repeat while(current->next!=NULL)

Current = current->next

[end loop]

Insert link at the end of list

[end of if structure]

[end of if structure]

Step 4: end

Algorithm for main program

Step 1: declare variables

Step 2: code for information page

Step 3: Take user input

Step 4: repeat for i=1 to 10

Call insert()

Print “complete list”

Print list

Print “List in reverse”

Call print\_backward()

Call split\_list()

Print “After splitting list”

Print “odd:”

Call printList()

Print “splitted list in reverse order”

Print “odd”

Call Print\_backward()

Print “ even”

Call print\_backward()

X: Print “Enter 0 to exit”

Input e

If(e=0) then

Exit

Else

Print “Enter the valid key”

goto x

Step 5: end

**Coding**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*prev;

struct node \*next;

};

struct node \*list = NULL;

struct node \*list\_last = NULL;

struct node \*even = NULL;

struct node \*even\_last = NULL;

struct node \*odd = NULL;

struct node \*odd\_last = NULL;

struct node \*current = NULL;

//Create Linked List

void insert(int data)

{

// Allocate memory for new node;

struct node \*link = (struct node\*) malloc(sizeof(struct node));

link->data = data;

link->prev = NULL;

link->next = NULL;

// If head is empty, create new list

if(list == NULL)

{

list = link;

return;

}

current = list;

// move to the end of the list

while(current->next!=NULL)

current = current->next;

// Insert link at the end of the list

current->next = link;

list\_last = link;

link->prev = current;

}

//display the list

void print\_backward(struct node \*head)

{

struct node \*ptr = head;

printf("\n[last] <=>");

//start from the beginning

while(ptr != NULL) {

printf(" %d <=>",ptr->data);

ptr = ptr->prev;

}

printf(" [head]\n");

}

//display the list

void printList(struct node \*head) {

struct node \*ptr = head;

printf("\n[head] <=>");

//start from the beginning

while(ptr != NULL) {

printf(" %d <=>",ptr->data);

ptr = ptr->next;

}

printf(" [last]\n");

}

void split\_list() {

// Allocate memory for new node;

struct node \*listp;

struct node \*link;

struct node \*current;

listp = list;

while(listp->next != NULL) {

struct node \*link = (struct node\*) malloc(sizeof(struct node));

link->data = listp->data;

link->prev = NULL;

link->next = NULL;

if(listp->data%2 == 0) {

if(even == NULL) {

even = link;

even\_last = link;

listp = listp->next;

continue;

} else {

current = even;

while(current->next != NULL) {

current = current->next;

}

// Insert link at the end of the list

current->next = link;

even\_last = link;

link->prev = current;

listp = listp->next;

}

} else {

if(odd == NULL) {

odd = link;

odd\_last = link;

listp = listp->next;

continue;

} else {

current = odd;

while(current->next!= NULL) {

current = current->next;

}

// Insert link at the end of the list

current->next = link;

odd\_last = link;

link->prev = current;

listp = listp->next;

}

}

}

// Lets handle the last node

if(listp!=NULL) {

link = (struct node\*) malloc(sizeof(struct node));

link->data = listp->data;

link->prev = NULL;

link->next = NULL;

if(listp->data%2 == 0) {

current = even;

while(current->next != NULL) {

current = current->next;

}

// Insert link at the end of the list

current->next = link;

even\_last = link;

link->prev = current;

} else {

current = odd;

while(current->next!= NULL) {

current = current->next;

}

// Insert link at the end of the list

current->next = link;

odd\_last = link;

link->prev = current;

}

}

}

int main()

{

int i,e,start,temp;

{

//clrscr();

printf("\n\n");

printf("\t\tShri Ramdeobaba College of Engineering and Management\n");

printf("\n\n\t\t\t Project Assesment on\n");

printf("\t\t Application to split doubly linked list into 2 doubly linked list\n");

printf("\n\t\t\t for the assesment of\n");

printf("\t\t\t MCA-I First Semester (Shift-I)\n");

printf("\n\t\t\t\t in the course\n");

printf("\t\t Concept of Data Structure Lab(MCP531)\n");

printf("\n\t\t\t\t developed by :\n");

printf("\t\t\t Vaibhav Shewale (Roll No 48)\n");

printf("\t\t\t Akanasha Bhoyar (Roll No 2)\n");

printf("\n\t\t\t under the guidance of\n");

printf("\t\t\t\tDr. Smita Nirkhi\n");

printf("\t\t\t Assestent Profrssor, RCOEM");

y: printf("\n\n\n\t\t\tEnter 1 to start the mini project : ");

scanf("%d",&start);

// getch();

}

if(start!=1)

{

printf("\t\t\tenter the correct choice:");

goto y;

}

else

{

system("cls");

for(i = 1; i <= 10; i++)

{

printf("enter the value for %d:",i);

scanf("%d",&temp);

insert(temp);

}

system("cls");

printf("Complete List : ");

printList(list);

printf("List in reverse: ");

print\_backward(list\_last);

split\_list();

printf("\nAfter splitting list - \n");

printf("Odd : ");

printList(odd);

printf("Even : ");

printList(even);

printf("\nSplitted lists in reverse - \n");

printf("Odd : ");

print\_backward(odd\_last);

printf("Even : ");

print\_backward(even\_last);

x:printf("Enter '0' to exit ");

scanf("%d",&e);

if(e==0)

exit(0);

else

{

printf("Enter the valid Key");

goto x;

}

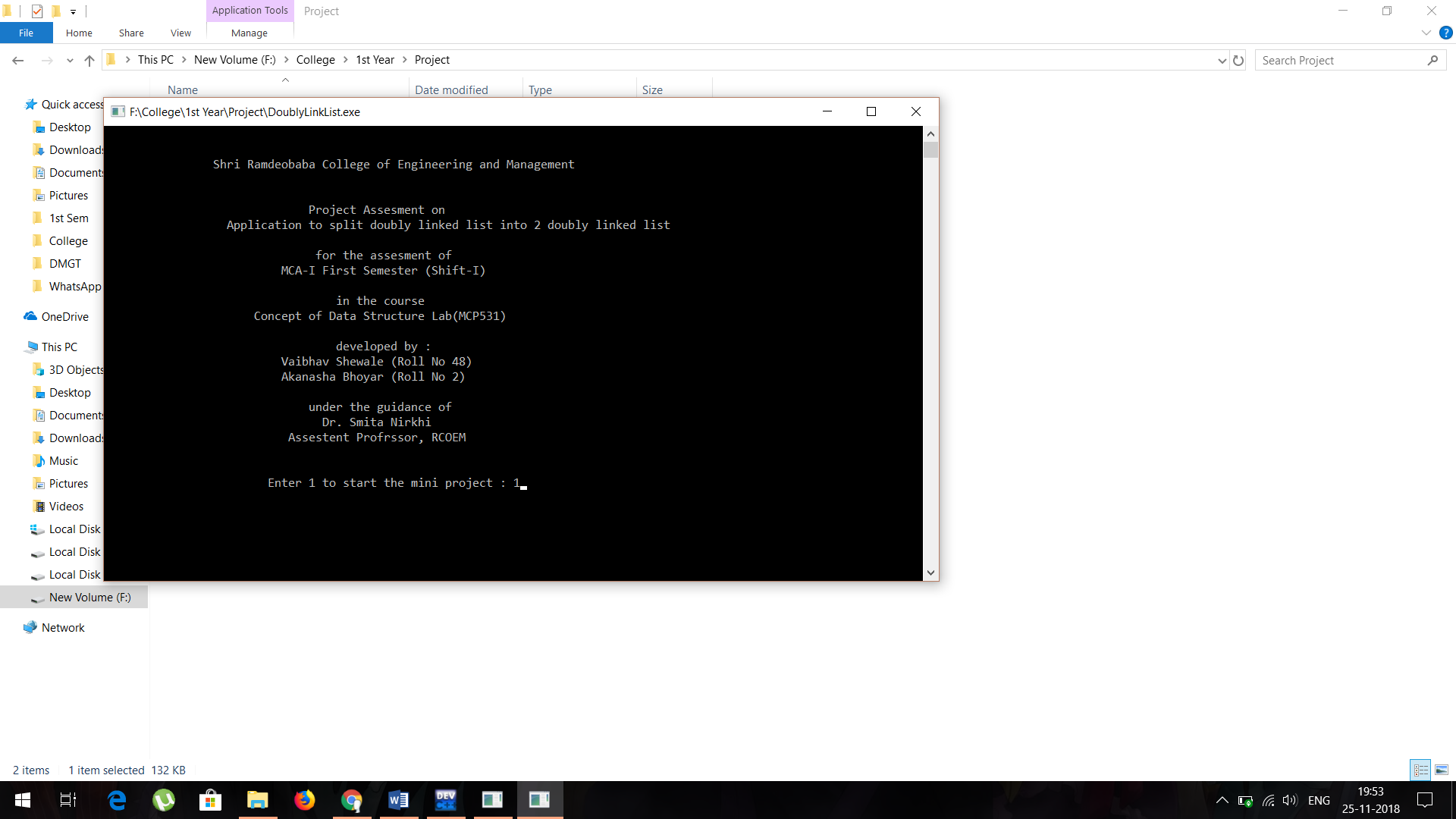
// return 0;

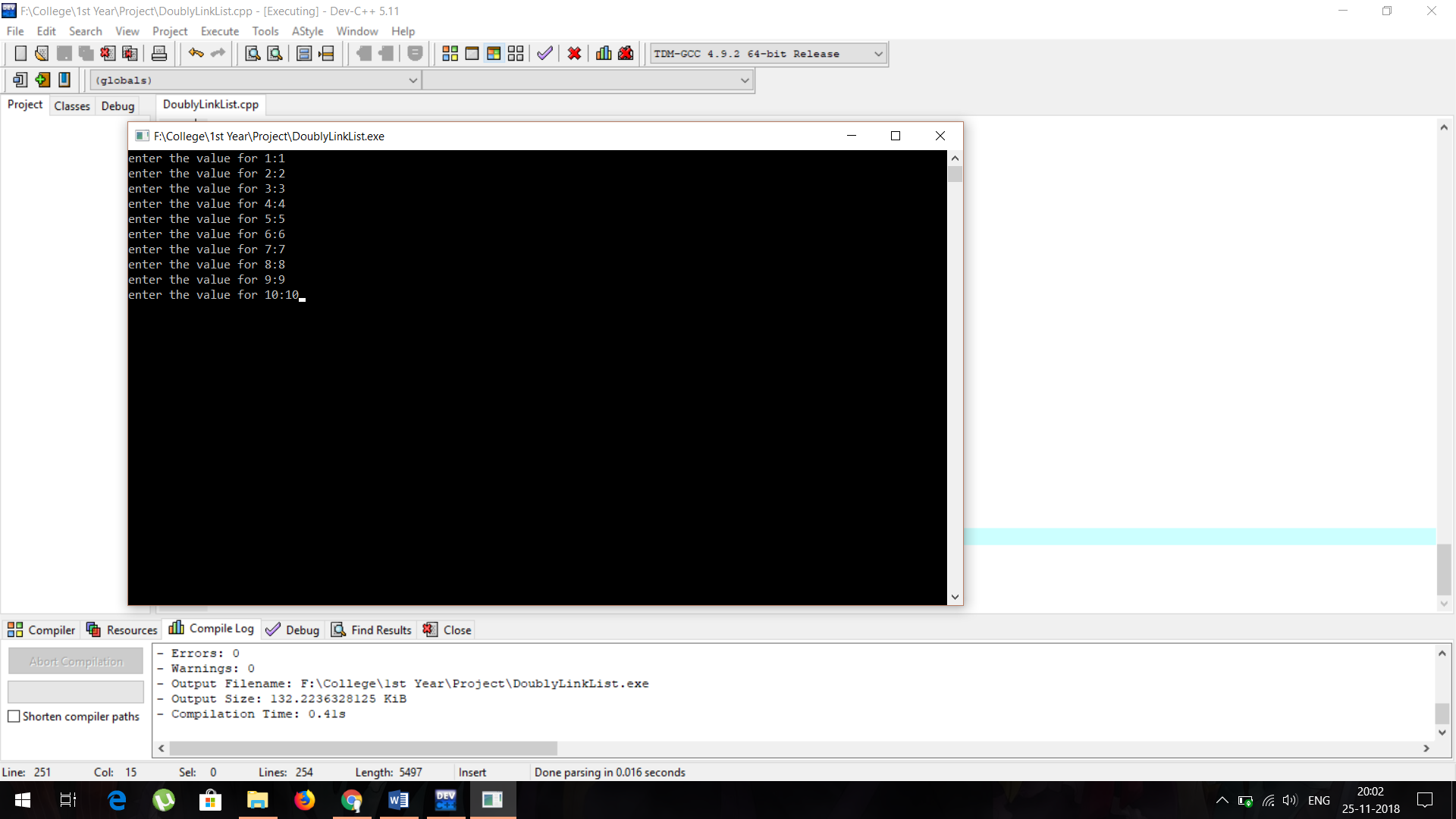
// getch();

}

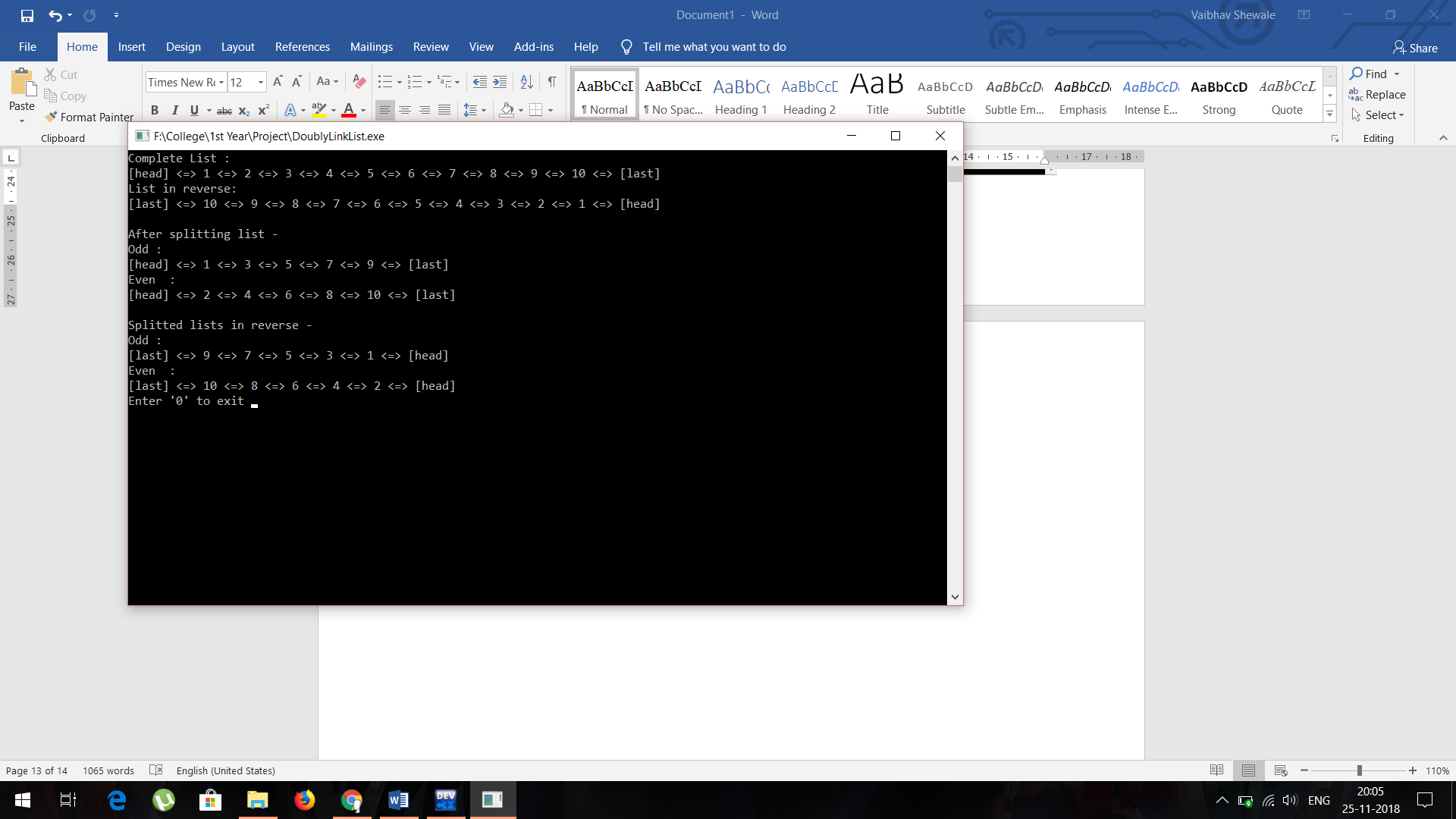
}

**Input Screens**





**Output Screens**



**Conclusion:** Thus, we successfully study & performed program on application to split doubly linked list into 2 doubly linked list and learn how to use doubly link list properly.