Programming assignment Staque Saba Konjaria

Code Description

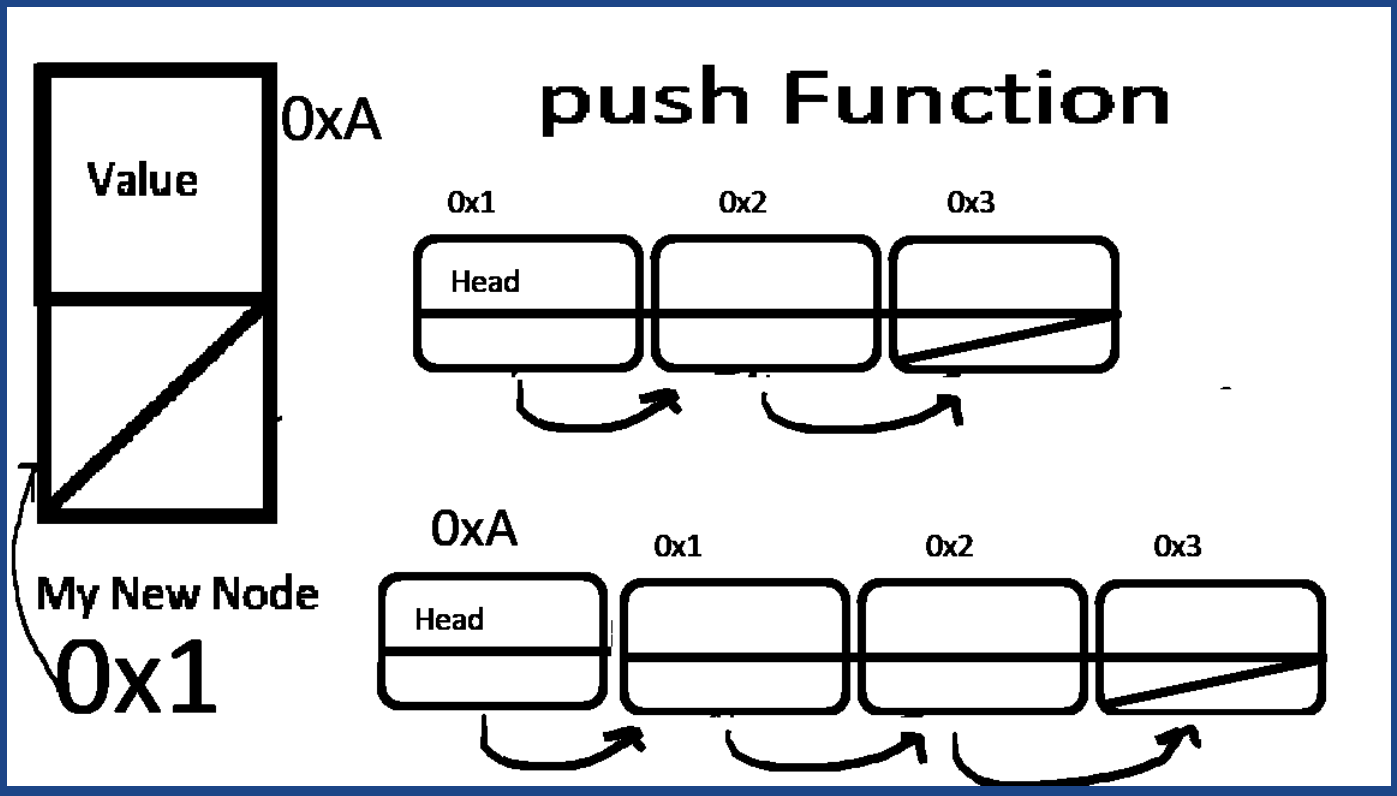
I created a Data structure called Staque , where you can pass and remove the integer numbers with helping of singly(forward) linked list implementation.I have used one (header) library function for input/output stream and as the values that could be inserted in the staque nodes are defined as int , in the beginning since i have said that type definition of integer could be used as MyType as well , so even when i wrote Mytype in the code that was also the same as int in this particular case. Then inside the class i have two access specifiers : private and public.

. . . *In private access specifier* i have used member variables for the size of my list so (numOfElements) and pointer to a class Node which holds the address of the top element. Class Node is also declared into my staque private section say so ,may you ask me what is this Node? Why it is so attractive to be declared in my Staque private access specifier? And i'll simply answer to this question that this is the class created for the elements of my staque , that should have their value and necessarily know the address of the next element , Since it has a NODE pointer as a data member in its public access specifier(in order not to have any problems in the Staque member functions where i am going to use of course those data members for sufficient functionalities) and from the definition of the pointer it should point to the address of something - so in this case pointer to a class Node , holds the address of the next element of the staque and here we are going to target singly linked list implementation. In Node constructor by default node has value that is 0 and next* gets the value *nullptr* .

. . . *And in Public access specifyier* In the beginning i have a Staque default value constructor that creates an empty Node, then there you can see all the functionalities that are useful for dynamic memory allocation that i am going to use shortly after.and those are copy constructor , destructor and overloaded assignment operator copy constructor and assignment overloading operator job is kinda the same ,but the difference between them is that copy constructor is invoked iff when we're going to assign a new Staque to another already created , so pass the data into it ,from another node Staque object. Overloading operator can be used after declaration and it is the general purpose of using this functionality. And of course we need destructor in order to delete say so , deallocate already allocated memory.
In push function i have similar idea planted - If numeric parameter that you are trying to store in the Staque is even, it is pushed in front else it will be pushed(inserted) back.

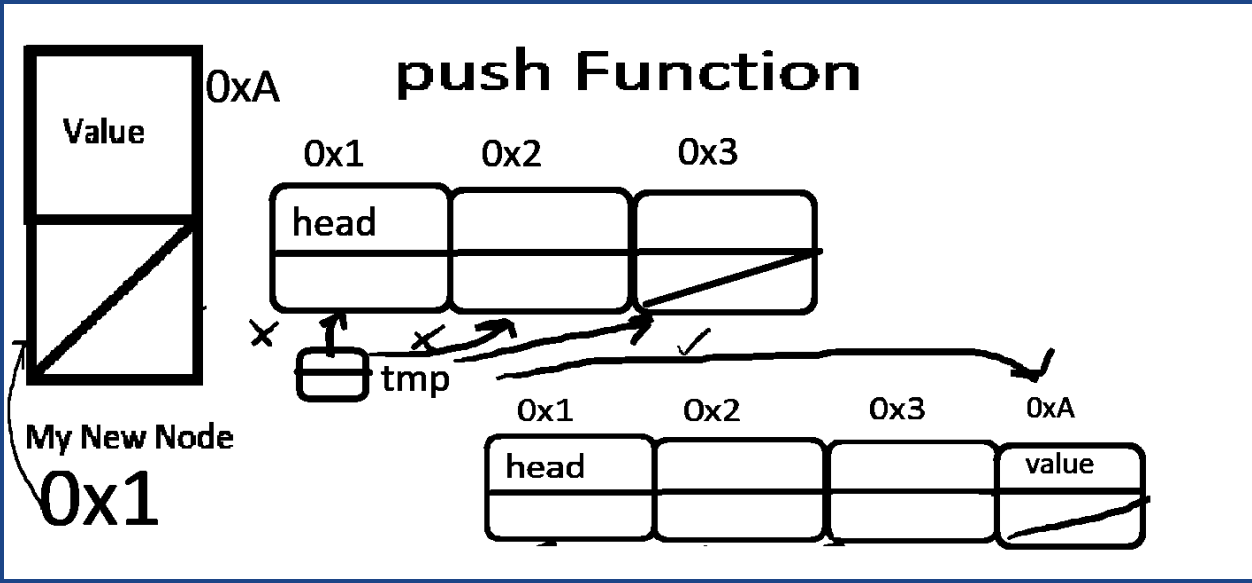
Insertion of even number

Basic idea of insertion Case: pushing even number first of all I am checking if the staque already that exists is empty or not and if it is empty to the very top element head* will hold the address of my new node that you can see on the picture with its provided value as a parameter and pointer to the next element is nullptr.but in else cases it is more complicated i'd say a little bit complicated i have still the same node but shortly after the compiler checked that there exist other elements in my staque already ,i need to say that pointer to a node class , (next*) should hold the address of already existed very top element (head)



Insertion of odd number

But in case of odd number insertion there happens the similar , I have the same new node since , i am going to check whether already existed node has elements or not if it has no elements I would have just one node that i created (0xA) but else , I have to create another node temporary pointer iteratively access the last element thats pointer to the next element is nullptr if this condition met the iteration stops and (0xA) will be passed back. Basic idea is also shown down below



In both cases the size of my Staque so the number of elements increases
For the remove function i have planted the same idea but also there is one thing ,how to delete with LIFO (last it first out) rule.Issue was fixed like that in my code I am going to pass the parameter and method should itself denote whether it is even number or not and it should made implementation respectively but there is one thing that if staque is not empty and there is not any even number here ,and the user provides to delete even number i took into the consideration that part as well and finally i decided to if it is not possible to delete even number from staque then let's delete odd number , since the user provides to delete / remove in general purpose.also I am showing the user the result variable that is useful for outputting the element that was removed already after invoking this functionality.
For error checking and validation i took into account that is there is nothing to delete so if the staque is empty the programme mustn't be able to delete anything because it is not possible so I am telling the user that it is impossible to implementate in user-friendly ways i hope.also of course I am decreasing the size of my Staque as well after deletion precesses.
Than comes method called size - i would say nothing special just number of elements planted into my staque is returning with invoking this functionality.

Traverse function - is useful to output on the screen details about my staque , so , all the nodes of the staque , implemented with forward_list(singly linked list)here i have also the possibility to use ostream(output stream object) for example std::cout i can use it in overloading ostream operator function as well , and i have done that.I really tried to output in very user-friendly way the data known about specific Staques. And way i am going to traverse all the nodes is to create a temporary pointer to a node class which holds the address of head, then iteratively output all the elements if there exists elements at all of course.

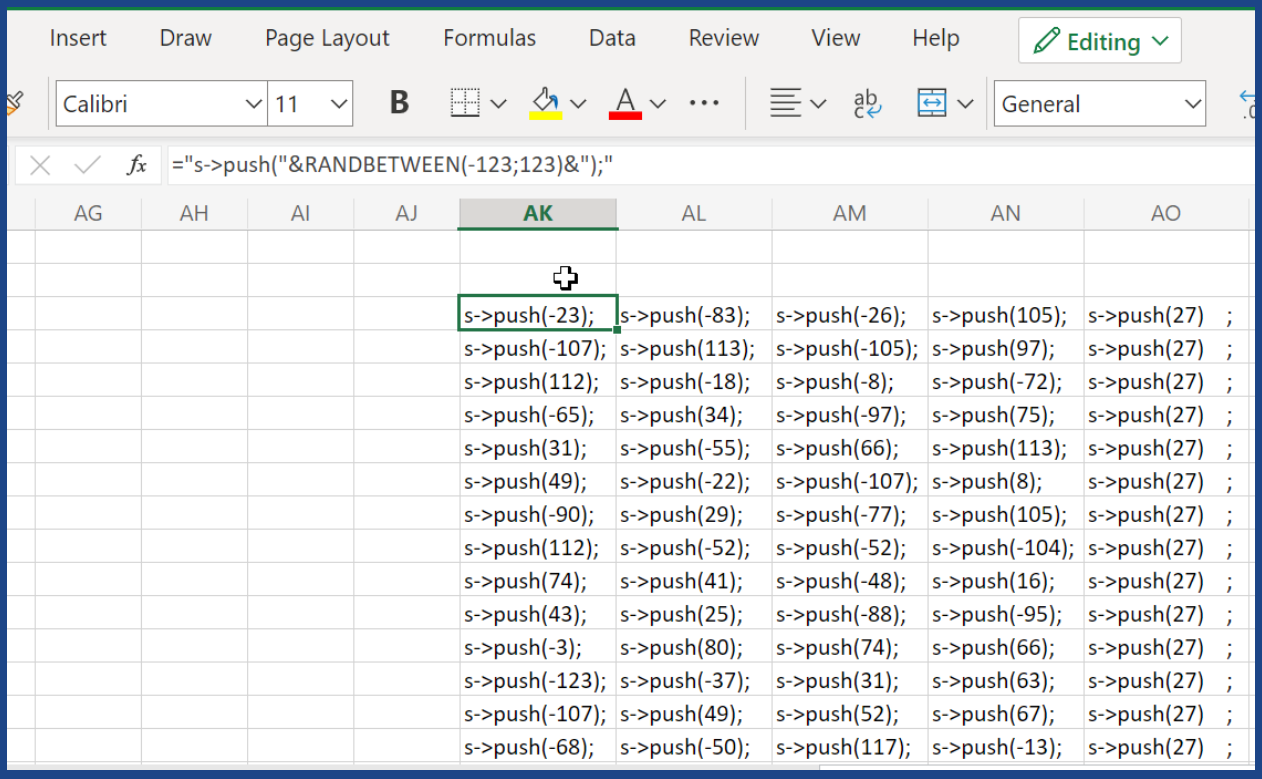
Basic Functions I have

- Push (insertion elements)
- Remove (deletion of elements)
- Traverse (display data)
- Size (returns the number of elements in the staque)
- isEmpty (returns True if the stack has no element else returns false)
- Yes_or_No (actually i could use boolalpha , but for user-friendly interface. . .)
- constructor (constructs the class objects)
- Destructor (here for destroying allocated memory)
- Copy constructor (take the data from one staque , and pass into another new one)
- Overloaded Assignment Operator (overloading assignment operator also does the same thing as copy constructor but say so it is more generic and i can invoke this function always ,rather than copy constructor , in order to copy the data say so from one object and paste into another)
- Overloaded ostream(output stream) operator << (gives me the possibility to traverse the elements of the staque with just writing of std::cout<<Staque_object;(if we take into account that Staque_object is already declared as object of the class called staque respectively).

Programming assignment Staque Saba Konjaria

For code Optimization . . .

In the Test 4 I have the similar- checking for a lot of data , insertion and deletion of course i had not to write a lot for it manually and loss a lot of data as well,so with helping of one of the best programme for such things and programmers in general I used the function that automatically traverse invoking of a function itself concatenates the string(s)(in order to invoke the function choose the correct syntaxes and etc.) and from the specific numeric range randomly passes the parameters in it shortly after i have created this function i dragged and dropped it, Finally cope and paste into my Visual studio code . For not wasting the time and also for writing as less lines of code as possible to do and also for understanding the programm functionalities well i chose this way , which helps me to instead of thinking and writing a lot about the passing parameters spend less than 1 minute.



AG	AH	AI	AJ	AK	AL	AM	AN	AO
				s->push(-23);	s->push(-83);	s->push(-26);	s->push(105);	s->push(27) ;
				s->push(-107);	s->push(113);	s->push(-105);	s->push(97);	s->push(27) ;
				s->push(112);	s->push(-18);	s->push(-8);	s->push(-72);	s->push(27) ;
				s->push(-65);	s->push(34);	s->push(-97);	s->push(75);	s->push(27) ;
				s->push(31);	s->push(-55);	s->push(66);	s->push(113);	s->push(27) ;
				s->push(49);	s->push(-22);	s->push(-107);	s->push(8);	s->push(27) ;
				s->push(-90);	s->push(29);	s->push(-77);	s->push(105);	s->push(27) ;
				s->push(112);	s->push(-52);	s->push(-52);	s->push(-104);	s->push(27) ;
				s->push(74);	s->push(41);	s->push(-48);	s->push(16);	s->push(27) ;
				s->push(43);	s->push(25);	s->push(-88);	s->push(-95);	s->push(27) ;
				s->push(-3);	s->push(80);	s->push(74);	s->push(66);	s->push(27) ;
				s->push(-123);	s->push(-37);	s->push(31);	s->push(63);	s->push(27) ;
				s->push(-107);	s->push(49);	s->push(52);	s->push(67);	s->push(27) ;
				s->push(-68);	s->push(-50);	s->push(117);	s->push(-13);	s->push(27) ;

Several run-time screenshots

First testing results

```
                                          Test 1  Results
*******************************************************
What about the staque?      : Empty
Stack size at the moment    : 0
Traverse the stack nodes    : 0 elements found
*******************************************************
 ---------   After insertion of elements  --------
Stack is empty ?  Answer    :  already Non-empty
Stack size at the moment    : 7
Traverse the stack nodes    : 7 elements found

         Staque  elements
        #1  : 8
        #2  : 6
        #3  : 4
        #4  : 2
        #5  : 1
        #6  : 3
        #7  : 9

*******************************************************
Removed  : 8
Removed  : 6
Removed  : 9

 Result after removing some elements from the staque
Stack is empty ?  Answer    : No
Stack size at the moment    : 4
Traverse the stack nodes    : 4 elements found

         Staque  elements
        #1  : 4
        #2  : 2
        #3  : 1
        #4  : 3
```

Test 2 Results

```
*********************************************************

                                          Test 2  Results
*********************************************************
What about the staque?       : Empty
Stack size at the moment     : 0
Traverse the stack nodes     : 0 elements found
*********************************************************
After insertion of elements into already Non-empty staque
Stack is empty ?  Answer     : No
Stack size at the moment     : 13
Traverse the stack nodes     : 13 elements found

         Staque  elements
        #1   : 500
        #2   : 0
        #3   : 1500
        #4   : -4
        #5   : -2
        #6   : 0
        #7   : 8
        #8   : 4
        #9   : 9
        #10   : -1
        #11   : -3
        #12   : -5
        #13   : -7

*********************************************************
```

```
**********************************************************
Removed   : 500
Removed   : -7
Removed   : -5
Removed   : 0
Removed   : -3
Removed   : -1
Removed   : 1500
After deletion of some  elements into the staque
Stack is empty ?  Answer    : No
Stack size at the moment    : 6
Traverse the stack nodes    : 6 elements found

         Staque  elements
        #1   : -4
        #2   : -2
        #3   : 0
        #4   : 8
        #5   : 4
        #6   : 9

**********************************************************
```

```
**********************************************************

Staque S2

Stack is empty ?  Answer    : No
Stack size at the moment    : 6
Traverse the stack nodes    : 6 elements found

         Staque  elements
        #1  : -4
        #2  : -2
        #3  : 0
        #4  : 8
        #5  : 4
        #6  : 9

**********************************************************



another one



Stack is empty ?  Answer    : No
Stack size at the moment    : 6
Traverse the stack nodes    : 6 elements found

         Staque  elements
        #1  : -4
        #2  : -2
        #3  : 0
        #4  : 8
        #5  : 4
        #6  : 9

**********************************************************
```

Result of overloaded assignment operator function and copy constructors , that takes the data from one object(pointer to a class object) and pass into another hown there

```
**********************************************************
After pushing some elements into an empty staque
Stack is empty ?  Answer     : No
Stack size at the moment     : 6
Traverse the stack nodes     : 6 elements found

         Staque  elements
        #1   : 30
        #2   : 0
        #3   : 30
        #4   : 3
        #5   : 13
        #6   : 3

**********************************************************
Removed   : 3
Removed   : 13
Removed   : 30


s3 elements


Stack is empty ?  Answer     : No
Stack size at the moment     : 3
Traverse the stack nodes     : 3 elements found

         Staque  elements
        #1   : 0
        #2   : 30
        #3   : 3

**********************************************************
```

```
another another one



Stack is empty ?  Answer     : No
Stack size at the moment     : 3
Traverse the stack nodes     : 3 elements found

         Staque  elements
        #1   : 0
        #2   : 30
        #3   : 3

**************************************************
Removed   : 0
Removed   : 30
Removed   : 3
What about the staque?       : Empty
Stack size at the moment     : 0
Traverse the stack nodes     : 0 elements found
**************************************************
```

Test 3 Results

```
Traverse the stack nodes    : 0 elements found
**************************************************

                                    Test 3  Results
**************************************************
what about the staque ?   - Staque is Empty

after pushing some elements in the staque
Stack is empty ?  Answer    :  already Non-empty
Stack size at the moment    : 5
Traverse the stack nodes    : 5 elements found

         Staque  elements
        #1  : 0
        #2  : 12
        #3  : 7
        #4  : 7
        #5  : 15

**************************************************
 Okay let's relax a little bit and i would like to beg you to provide some numbers here , but please only numbers

Number 1 :
```

Let’s make my programme user-defined , i am going to ask the user to provide two numbers , and with helping og logical , unary , bitwise and etc operators i am pushing the numbers in the list of the elements of the staque, for those operators ,i don’t want the second number to be zero in order to avoid this staff i am asking as many times as is possible ,to input correctly.

```
**************************************************
 Okay let's relax a little bit and i would like to beg you to provide some numbers here , but please only numbers

Number 1 : 1

Number 2 : saba
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : konjaria
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : are you okay?
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : 0
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : no  i am not
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : 0
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : 0
so far i am going to use some operatiosn please provide anything else besides to zero anymore

Number 2 : 1
```

```
Number 2 : 1
1 and 1 will be inserted into a(n)Non-empty s1 staque

Result we get
Stack is empty ?  Answer    : No
Stack size at the moment    : 7
Traverse the stack nodes    : 7 elements found

         Staque  elements
        #1  : 0
        #2  : 12
        #3  : 7
        #4  : 7
        #5  : 15
        #6  : 1
        #7  : 1

**********************************************************
I  am going to insert those values with helping of bitwise , logical , arithmethic,
```

```
**************************************************
I  am going to insert those values with helping of bitwise , logical , arithmethic, unary  and  etc operators
After insertion   Stack is empty ?  Answer    : No
Stack size at the moment    : 19
Traverse the stack nodes    : 19 elements found

        Staque  elements
       #1  : 0
       #2  : 0
       #3  : 2
       #4  : 0
       #5  : 2
       #6  : 0
       #7  : 0
       #8  : 0
       #9  : 12
       #10  : 7
       #11  : 7
       #12  : 15
       #13  : 1
       #14  : 1
       #15  : 1
       #16  : 1
       #17  : 1
       #18  : 1
       #19  : 1

**************************************************
```

```
**************************************************
Removed  : 0
Removed  : 1
Removed  : 1
Removed  : 1
Removed  : 1
Removed  : 1
Removed  : 1
Removed  : 1
Removed  : 15
Removed  : 7
Removed  : 7
Removed  : 12
Removed  : 0
Removed  : 0
After deletion    Stack is empty ?  Answer    : No
Stack size at the moment    : 5
Traverse the stack nodes    : 5 elements found

         Staque  elements
        #1  : 2
        #2  : 0
        #3  : 2
        #4  : 0
        #5  : 0

**************************************************
```

Test 4 Results

Here I am checking if a lot of data for my staque (i am pushing about 307 elements in my staque) and check if there is some issue ,with it after increasing amount of elements as much

```
*****************************************************

                                    Test 4  Results
*****************************************************
in the beginning of course   Staque called s  is  Empty
Staque (s) 's size   : 0

 * ****************  checking for a lot of data  ********************************************
After pushing a lot of the data ,my staque looks like this
Stack is empty ?  Answer    : No
Stack size at the moment    : 307
Traverse the stack nodes    : 307 elements found

         Staque  elements
        #1  : -94
        #2  : 0
        #3  : -102
        #4  : 12
        #5  : 56
        #6  : -40
        #7  : 118
        #8  : 52
        #9  : -96
        #10  : 68
        #11  : -66
        #12  : 44
```

```
        #282  : 17
        #283  : -19
        #284  : 85
        #285  : -87
        #286  : 65
        #287  : 13
        #288  : -67
        #289  : -79
        #290  : -43
        #291  : 117
        #292  : 89
        #293  : -19
        #294  : -71
        #295  : -47
        #296  : -111
        #297  : 57
        #298  : 87
        #299  : 65
        #300  : 95
        #301  : -9
        #302  : -43
        #303  : 91
        #304  : -25
        #305  : 103
        #306  : -69
        #307  : -27

*********************************************************
```

```
**************************************************
Removed  : -94
Removed  : 0
Removed  : -102
Removed  : 12
Removed  : 56
Removed  : -40
Removed  : 118
Removed  : 52
Removed  : -96
Removed  : 68
Removed  : -66
Removed  : 44
Removed  : -56
Removed  : 90
Removed  : -58
Removed  : -80
Removed  : 114
Removed  : 62
Removed  : 96
Removed  : -44
Removed  : -10
Removed  : 120
Removed  : 28
Removed  : 120
Removed  : -54
Removed  : 30
Removed  : -76
Removed  : 6
  After deletion of a lot of even numbers from  my staque elements looks like this
Stack is empty ?  Answer    : No
Stack size at the moment    : 279
Traverse the stack nodes    : 279 elements found
```

```
        #254  : 17
        #255  : -19
        #256  : 85
        #257  : -87
        #258  : 65
        #259  : 13
        #260  : -67
        #261  : -79
        #262  : -43
        #263  : 117
        #264  : 89
        #265  : -19
        #266  : -71
        #267  : -47
        #268  : -111
        #269  : 57
        #270  : 87
        #271  : 65
        #272  : 95
        #273  : -9
        #274  : -43
        #275  : 91
        #276  : -25
        #277  : 103
        #278  : -69
        #279  : -27

**********************************************************
**********************************************************
That's all have a good day , I hope we meet each other again
```

Code in C++

```
#include <iostream> //library function

typedef int Mytype;

//definition of Mytype is int  at this particular moment
                                                        //howevere i can modify it as many times
as i want to and also ,
                                                    // easily i can manipulate with stack as well
with another types of data
//preprocessors
//      Yes_or_No(tmp != nullptr, 4, this->numOfElements);

void  Yes_or_No(bool condition, int which_one = 1, int size = 0) {
        switch (which_one)
        {
        case 1: (condition) ? std::cout << "Yes" : std::cout<<"No"; break;
        case 2:  (condition) ? std::cout<<"Empty" : std::cout<<"Non-empty";
break;//especially useful for isempty() condition
        case 3:  (condition) ? std::cout<<" still Empty" : std::cout<<" already Non-empty";
break;//especially useful for isempty() condition
        case 4:  (condition) ? std::cout << size << " elements found" : std::cout<<size << "
elements found"; break;
        default:
                break;
        }

}

class Staque {
        friend std::ostream& operator<<(std::ostream&, Staque&); //Staque's friend
overloading output stream operator function
private:
        class Node { //class node data members in Staques private access modifyer
        public:
                Mytype value;  // node is the element of a staque , that has a value
                Node* next;   // and node knows also , the next element of my data structure
                Node( Mytype value = 0, Node* next = NULL) {
                        this->value = value;
                        this->next = next;
                }  // constructor for staque
                //since  programmers are lazy people i wrote both explicit and default
constr.s in one way
```

```
        };
        int numOfElements;
        Node* head;                 //particularly data members of staque's object

public:

//deafult value constructor  for staque that creates an empty stack ( have not anydata in
its node )

        Staque();

//                      copy constructor
//
//   Example :                  Staque* s1= new Staque;
//                                          s1->push(3);
//                                          s1->push(3);
//                                          s1->push(3);
//                                          Staque* s2 = s1;
//                                          s2->traverse(std::cout);
//
// Expectable output :  3 3 3

        Staque(const Staque& other) {
                head = NULL;
                if (!other.head == 0)
                {
                        // Copy first node
                        head = new Staque::Node(other.head->value);
                        // Set pointers to run through the stacks' linked lists
                        Staque::Node* lastPtr = head;
                        Staque::Node* origPtr = other.head->next;
                        while (origPtr != 0)
                        {
                                lastPtr->next = new Staque::Node(origPtr->value);

                                lastPtr = lastPtr->next;
                                origPtr = origPtr->next;
                        }
                }
        }
//  Destructor

        ~Staque();
```

```
// Overloading assignment operator
//        Example :      Staque* s3, * s4, * s5;
//                                    s3 = s4;

        const Staque& operator= (const Staque& rightHandSide);

//function for inserting

        void push(Mytype value);

// Deletion of even number from the staque

        void remove(int number_to_delete);

// Deletion of odd number from the staque


// returns the size of my  staque

        int size();

// return T( true ) if stack's top element , is  nullpointer  else F ( false )

        bool isEmpty() {
                return head == NULL;
        }

//function to traverse the data planted into the staque

        void traverse(std::ostream& COUT, int value);


};

//Definition of Staque constructor ( creation of empty node )

Staque::Staque() {
        head = NULL;
        numOfElements = 0;
}

//Definition of Staque destructor  ( qith helping of head , which is the accessor of the top
element in the node
//it is checking if there is some data planted and deletes respectively
```

```
Staque :: ~Staque() {
	while (head != NULL) {
		Node* next = head->next;
		delete head;
		head = next;
	}
	//if  head == is true, while loop can't be occured
}

// definition of overloading assignment operator

const Staque& Staque::operator=(const Staque& rightHandSide)
{
	if (this != &rightHandSide)
	{
		this-> ~Staque();
		if (rightHandSide.numOfElements==0)
			head = nullptr;
		else
		{
			// check that not st st
			// destroy current linked list
			// empty stack
			// copy rightHandSide's list
			// Copy first node
				head = new Staque:: Node(rightHandSide.head->value);
				// Set pointers to run through the staques linked lists
				Staque::Node* lastPtr = head;
				Staque::Node* rhsPtr = rightHandSide.head->next;
					while (rhsPtr != nullptr)
					{
						lastPtr->next = new
Staque::Node(rhsPtr->value);
						lastPtr = lastPtr->next;
						rhsPtr = rhsPtr->next;
					}
		}
	}
	return *this;

}

//Insert element in Staque
```

```
void Staque::push(Mytype value) {
	Node* current = new Node(value, nullptr);  // node has top element assigned to nothing (null) and its value is something that is
// passed as a parameter and after that ther will be nothing again( null ) so this value is somewhere in the middle
//so  it is the only one element of this node as well since next or top elements are nullpointers
//If the number that you are trying to store in the Staque is even,
//it is pushed in front of the Staque
	if (value % 2 == 0) {
		if (head == NULL) {
// first checking If stack is empty
			head = current;
		}
		else {
			current->next = head; //in front of the staque
			head = current;
		}
	}
	//If the number that you are trying to store in the Staque is odd,
//it is pushed at the end of the Staque
	else {
		if (head == NULL) { //If stack is empty again we need to be ensure that this thing won't affect our programme
			head = current;
		}
		else {
			Node* tmp = head;
			while (tmp->next != NULL) { //access the  data from the ending point
				tmp = tmp->next;
			}
			tmp->next = current;
		}
	}
	numOfElements++; //Increase Number of elements
}

//Remove  value from front if one we want to remove is even number   else delete from back

void Staque::remove(int number_to_delete) {
	if (number_to_delete % 2 == 0) {
		if (head == NULL) {
```

```
                std::cout << "Stack is Empty, Can't remove" << std::endl;
                return;
        } //if there is nothing into my staque i can not delete anything
        Node* temp = this->head;

        Mytype result = head->value;
        Node* next = head->next;
        delete head;
        head = next;
        numOfElements--; //Decrease Number of elements
        std::cout <<"Removed  : " << result << std::endl;
    }
    else {
        if (head == NULL) {
                std::cout << "Stack is Empty, Can't remove" << std::endl;
                return;
        }
        Mytype result;
        Node* tmp = head;
        if (tmp->next == NULL) {
                result = tmp->value;
                delete tmp;
                head = NULL;
                numOfElements--; //Decrease Number of elements
                std::cout <<"Removed  : " << result << std::endl;
                return;
        }
        while (tmp->next->next != NULL) { //Go at the end of the list
                tmp = tmp->next;
        }
        result = tmp->next->value;
        delete tmp->next;
        tmp->next = NULL;
        numOfElements--; //Decrease Number of elements
        std::cout<<"Removed  : " << result << std::endl;
    }
}

//Return size/number of elements

int Staque::size() {
    return numOfElements;
}
```

```
//Traverse the basic information about the list

void Staque::traverse(std::ostream& COUT,int value=1) {
	Node* tmp = head; // temporary node pointer points to the head element
	if (size() > 0) {
		using namespace std;
		cout
			<< "Stack is empty ?  Answer    : ";
Yes_or_No(this->isEmpty(),value);std::cout<<"\n"
			<< "Stack size at the moment    : " << this->size() << endl
			<< "Traverse the stack nodes    : "; Yes_or_No(tmp != nullptr, 4,
this->size()); std::cout<<std::endl;
		if (tmp != NULL) {
			std::cout << "\n\t Staque  elements  " << std::endl;
			while (tmp != NULL) {

				for (int i = 0; i < numOfElements; i++) {
					std::cout << "\t#" << i + 1 << "  : " << tmp->value <<
std::endl;
					tmp = tmp->next;
				}
				std::cout << std::endl;
			}
			std::cout << "*****************************************************" <<
std::endl;
		}
	}
	else {
		std::cout
			<< "What about the staque?              : ";
Yes_or_No(this->isEmpty(),value=2);
		std::cout << std::endl
			<< "Stack size at the moment    : " << this->size() << std::endl
			<< "Traverse the stack nodes    : "; Yes_or_No(tmp != nullptr, 4,
this->size()); std::cout << std::endl
			<< "*****************************************************" << std::endl;
	}


}


void Test1() {
```

```
        using namespace std;                                    //Create Staque object
        Staque s;
         s.traverse(cout);

        s.push(1);
        s.push(3);
        s.push(2);
        s.push(4);
        s.push(6);
        s.push(8);
        s.push(9);
        std::cout << " ---------   After insertion of elements  -------- " << std::endl;
                        //display list nodes  after insertion
         s.traverse(cout,3);
        //Remove some elements from the staque
        s.remove(2);
        s.remove(2);
        s.remove(3);
        //removing two even and one odd  numbers
        std::cout << "\n Result after removing some elements from the staque \n " << s ;

}

void Test2() {

        using namespace std;
        Staque* another_s2=new Staque;
        another_s2->traverse(cout);
        another_s2->push(4);//(even)
        another_s2->push(9);//(odd)
        another_s2->push(8);//(even)
        another_s2->push(0);//(even)
        another_s2->push(-2.f);//(even)
        another_s2->push(-4.f);//(even)
        another_s2->push(-1.f);//(odd)
        another_s2->push(-3);//(odd)
        another_s2->push(-5);//(odd)
        another_s2->push(-7);//(odd)
        another_s2->push(1.5e3);//1.5*10^3=1500(floating number ,converted to integer
number) (even)
        another_s2->push(0.5e-1);//0.5* 10^(-1) = 0 (int) (even)
        another_s2->push(0.5e3);//0.5* 10^3= 500 ( even )
```

```
        std::cout << "After insertion of elements into"; Yes_or_No(another_s2->isEmpty(),
3);std::cout << " staque " << std::endl<< *another_s2;
                                                        another_s2->remove(0);//removes
even number
                                                        another_s2->remove(1);//removes
odd number
                                                        another_s2->remove(1);//removes
odd number
                                                        another_s2->remove(4);//removes
even number
                                                        another_s2->remove(3);//removes
odd number
                                                        another_s2->remove(1);//removes
odd number
                                                        another_s2->remove(0);//removes
even number
std::cout << "After deletion of some  elements into the staque " << *another_s2 ;


        Staque* s2 = another_s2;    //cpy constr called


                cout << "\nStaque S2\n" << *s2;

                //expectable output has the same as another_s2->travers(cout) had


                                                        Staque* anotherone;
                                                        anotherone=s2;//overloaded
assignment  operators
                                                        cout  <<"\n\n\nanother
one\n\n\n"<<endl;

                anotherone->traverse(cout);

                                                        Staque* s3 = new Staque;
                                                        s3->push(30);
                                                        s3->push(3);
                                                        s3->push(0);
                                                        s3->push(13);
                                                        s3->push(30);
                                                        s3->push(3);
```

```
                                                        std::cout << "After pushing some
elements into an empty staque" << std::endl; s3->traverse(cout);
                                                        s3->remove(3);//removes odd
number from the staque
                                                        s3->remove(1);//removes odd
number from the staque
                                                        s3->remove(0);//removes even
number from the staque
                                                        anotherone = s3;
                                                        std::cout << "\n\ns3 elements\n\n"
<< std::endl; s3->traverse(cout);
                                                        std::cout << "\n\n\nanother
another one\n\n\n\n" << endl; anotherone->traverse(cout);
        anotherone->remove(2);
        anotherone->remove(2);
        anotherone->remove(1);

        anotherone->traverse(cout);

        //removes 2 even numbers and traverse the nodes  of linked list
}

void Test3() {

        Staque* s1=new Staque;
        Staque*  s2 = s1;
        int number1=0, number2=0;
        int* ptr1=&number1, * ptr2=&number2;
        std::cout << "what about the staque ?   - Staque is ";
Yes_or_No(s1->isEmpty(),2);std::cout << std::endl;
        s1->push(2*6);//12
        s1->push(0%2);//0
        s1->push(2+5);//7
        s1->push(3|4);//7
        s1->push(7|9|4);//15
        std::cout << "\nafter pushing some elements in the staque " << std::endl;
s1->traverse(std::cout,3);
        std::cout << " Okay let's relax a little bit and i would like to beg you to provide
some numbers here , but please only numbers" << std::endl;
        std::cout << "\nNumber 1 : "; std::cin >> number1;
        while (std::cin.fail()) {
                std::cin.clear();
                std::cin.ignore(100000, '\n');
                std::cout << "Incorrect input please provide only numbers " << std::endl;
```

```
                std::cout << "\nNumber 1 : "; std::cin >> *&number1;

        }
        std::cout << "\nNumber 2 : "; std::cin >> number2;
        while (std::cin.fail()  || number2==0) {
                std::cin.clear();
                std::cin.ignore(100000, '\n');
                if (number2 == 0) {
                        std::cout << "so far i am going to use some operatiosn please
provide anything else besides to zero anymore" << std::endl; std::cout << "\nNumber 2 :
"; std::cin >> *&number2;
                        continue;
                }
                std::cout << "Incorrect input please provide only numbers " << std::endl;
                std::cout << "\nNumber 2 : "; std::cin >> *&number2;

        }
        std::cout
                << *ptr1 << " and " << *ptr2 << " will be inserted into a(n)";
Yes_or_No(s1->isEmpty(), 2);std::cout<<" s1 staque  " << std::endl;
        s1->push(*ptr1);
        s1->push(*ptr2);
        std::cout << "\nResult we get " << std::endl;
        s1->traverse(std::cout);
        std::cout << "I  am going to insert those values with helping of bitwise , logical ,
arithmethic, unary  and  etc operators " << std::endl;
        // for provided input 1 and 1
        // 1 - 0001
        // 1 - 0001
        s1->push(number1 & number2);//1

        s1->push(number1 & number2);
        s1->push(number1 ^ number2);
        s1->push(number1 >> number2);
        s1->push(number1 << number2);
        s1->push(number1 | number2);
        s1->push(number1 % number2);
        s1->push(number1 + number2);
        s1->push(number1 - number2);
        s1->push(number1 * number2);
        s1->push(number1 == number2);
        s1->push(number1 != number2);
        std::cout << "After insertion   "; std::cout << *s1;
        s2->remove(0);//removes even number from my staque list of ellements
```

```
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(1);//removes odd number from my staque
	s2->remove(0);//removes even numbers from the staque list of elements
	std::cout << "After deletion    "<<*s1; //Test 2 final result

}

void Test4() {
	//let's check code sufficiency with a big aou

	Staque * s=new Staque;

	std::cout << "in the beginning of course   Staque called s  is  ";
Yes_or_No(s->isEmpty(), 2);
	std::cout<<"\nStaque (s) 's size   : " << s->size()<<std::endl;
	s->push(-2);    s->push(12); s->push(123);s->push(123); s->push(123);
s->push(123);s->push(123);


		s->push(-55); s->push(41);  s->push(-14); s->push(77);  s->push(-50);
		s->push(48);  s->push(-115);         s->push(64);  s->push(77);  s->push(-20);
		s->push(89);  s->push(34);  s->push(2);             s->push(117);
s->push(-107);
		s->push(-30); s->push(-69); s->push(-44); s->push(83);  s->push(105);
		s->push(19);  s->push(100);s->push(19);  s->push(-30); s->push(-113);
		s->push(76);  s->push(71);  s->push(107);s->push(-29); s->push(-82);
		s->push(-30); s->push(-41); s->push(-32); s->push(-31); s->push(-63);
		s->push(118);s->push(112);s->push(-10); s->push(56);  s->push(-115);
		s->push(-77); s->push(12);  s->push(-77); s->push(119);s->push(-33);
		s->push(108);s->push(15);  s->push(-91); s->push(-106);         s->push(-96);
		s->push(-16); s->push(-59); s->push(-104);         s->push(-38); s->push(66);
		s->push(-23); s->push(118);s->push(83);  s->push(-112);         s->push(119);
		s->push(64);  s->push(19);  s->push(-18); s->push(-85); s->push(-83);
		s->push(73);  s->push(111);s->push(-38); s->push(-44); s->push(45);
```

```
s->push(43);  s->push(-28); s->push(99);  s->push(-45); s->push(54);
s->push(-16); s->push(68);  s->push(-69); s->push(59);  s->push(-110);
s->push(-95); s->push(50);  s->push(-64); s->push(-86); s->push(-65);
s->push(24);  s->push(-4);  s->push(-94); s->push(-60); s->push(-123);
s->push(-91); s->push(-60); s->push(104);s->push(21);  s->push(-25);
s->push(-46); s->push(23);  s->push(11);  s->push(50);  s->push(-89);
s->push(-9);  s->push(18);  s->push(70);  s->push(69);  s->push(-47);
s->push(-6);  s->push(-101);        s->push(31);  s->push(-51); s->push(-6);
s->push(-105);        s->push(90);  s->push(-4);  s->push(-97); s->push(56);
s->push(-83); s->push(57);  s->push(17);  s->push(-119);        s->push(-16);
s->push(-13); s->push(-63); s->push(98);  s->push(-63); s->push(-18);
s->push(109);s->push(-55); s->push(-29); s->push(58);  s->push(109);
s->push(-7);  s->push(-67); s->push(-6);  s->push(-99); s->push(108);
s->push(-57); s->push(49);  s->push(-44); s->push(115);s->push(61);
s->push(-9);  s->push(36);  s->push(-8);  s->push(-18); s->push(82);
s->push(-45); s->push(49);  s->push(62);  s->push(120);s->push(-68);
s->push(66);  s->push(-113);        s->push(-76); s->push(-71); s->push(27);
s->push(22);  s->push(-12); s->push(-4);  s->push(-32); s->push(101);
s->push(-97); s->push(30);  s->push(-102);        s->push(33);  s->push(-28);
s->push(80);  s->push(36);  s->push(60);  s->push(40);  s->push(-59);
s->push(-60); s->push(109);s->push(-3);  s->push(48);  s->push(-121);
s->push(-5);  s->push(115);s->push(9);      s->push(-54);       s->push(57);
s->push(-112);        s->push(-113);        s->push(-102);        s->push(121);
s->push(-123);
s->push(-66); s->push(-112);        s->push(33);  s->push(-16);
s->push(-100);
s->push(-113);        s->push(72);  s->push(-69); s->push(100);s->push(76);
s->push(-98); s->push(-97); s->push(68);  s->push(-5);  s->push(-69);
s->push(70);  s->push(-81); s->push(-28); s->push(-89); s->push(116);
s->push(-53); s->push(98);  s->push(90);  s->push(-12); s->push(-86);
s->push(-68); s->push(72);  s->push(73);  s->push(53);  s->push(55);
s->push(0);           s->push(-21); s->push(67);  s->push(32);  s->push(-27);
s->push(112);s->push(-47); s->push(89);  s->push(-16); s->push(-87);
s->push(-67); s->push(58);  s->push(93);  s->push(93);  s->push(118);
s->push(-49); s->push(-110);        s->push(33);  s->push(22);  s->push(10);
s->push(-14); s->push(-44); s->push(113);s->push(6);           s->push(-33);
s->push(-107);        s->push(-76); s->push(30);  s->push(95);  s->push(-54);
s->push(61);  s->push(120);s->push(-71); s->push(-23); s->push(47);
s->push(51);  s->push(28);  s->push(120);s->push(17);  s->push(-19);
s->push(-10); s->push(85);  s->push(-87); s->push(65);  s->push(13);
s->push(-67); s->push(-44); s->push(96);  s->push(62);  s->push(114);
s->push(-79); s->push(-43); s->push(117);s->push(-80); s->push(89);
s->push(-19); s->push(-71); s->push(-58); s->push(-47); s->push(90);
s->push(-111);        s->push(57);  s->push(-56); s->push(87);  s->push(44);
```

```
                s->push(-66); s->push(65);  s->push(68);  s->push(95);  s->push(-9);
                s->push(-43); s->push(91);  s->push(-96); s->push(52);  s->push(118);
                s->push(-40); s->push(-25); s->push(56);  s->push(12);  s->push(103);
                s->push(-102);         s->push(0);              s->push(-69); s->push(-94);
s->push(-27);

                        std::cout << "\n * ****************  checking for a lot of data
*********************************************" << std::endl
                         << "After pushing a lot of the data ,my staque looks like this  " <<
std::endl;
        s->traverse(std::cout);

        s->remove(0); s->remove(0); s->remove(0); s->remove(0);
        s->remove(0); s->remove(0); s->remove(0); s->remove(0);
        s->remove(0); s->remove(0); s->remove(0); s->remove(0);
        s->remove(0); s->remove(0); s->remove(0); s->remove(0);
        s->remove(0); s->remove(0); s->remove(0); s->remove(0);
        s->remove(0); s->remove(0); s->remove(0); s->remove(0);
        s->remove(0); s->remove(0); s->remove(0); s->remove(0);

        std::cout << "  After deletion of a lot of even numbers from  my staque elements
looks like this " << std::endl;

        s->traverse(std::cout);

        //if there is not even numbers anymoreinto my staque , my class member function
called
        //remove() , removes by itself , odd number , that means that i am  going to delete
either even or odd numbers from the staque
        //depending on the situation
}

int main() {
        //Remove 0 means stack is empty and can't remove
        std::cout << "                                Test 1  Results                                "
<< std::endl;
        std::cout << "*******************************************************" << std::endl;
        Test1();
        std::cout << std::endl;
        std::cout << "                                Test 2  Results                                "
<< std::endl;
        std::cout << "*******************************************************" << std::endl;
                Test2();
        std::cout << std::endl;
```

```
	std::cout << "                    Test 3  Results                    "
<< std::endl;
	std::cout << "******************************************************" << std::endl;
	Test3();
	std::cout << std::endl;
	std::cout << "                    Test 4  Results                    "
<< std::endl;
	std::cout << "******************************************************" << std::endl;
	Test4();
	std::cout << "******************************************************" << std::endl;
	std::cout << "That's all have a good day , I hope we meet each other again " <<
std::endl;

	system("pause");
	return 0;
}

//		definition of overloading output stream operator

std::ostream& operator<<(std::ostream& COUT, Staque& obj)
{
	obj.traverse(COUT);
	return COUT;
}
```