|  |
| --- |
| **Data Structures and Algorithms**  ***Section*: BSCE2021 Assignment # 6 *Total marks*: 100**  ***Name*** : ***\_nimra maqbool\_ Roll number* : \_bsce21012\_** |

***Submission:***

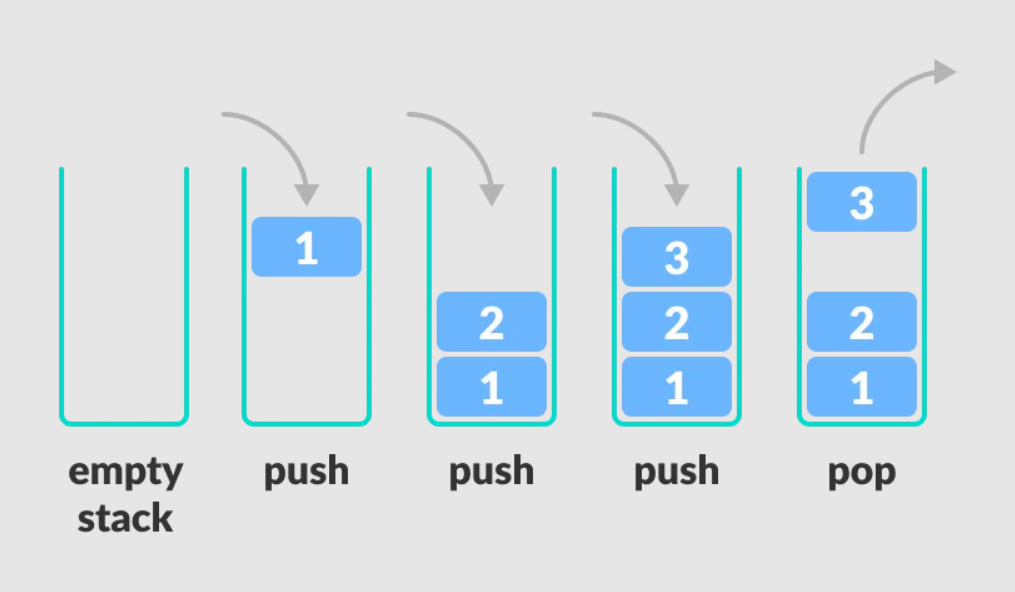
• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.*

*• Submission after due time will not be accepted.*

**There should be a Report explaining your code and highlighting the results. Follow this naming convention for your report RollNumber\_Assignment#.pdf e.g BSCE21001\_Assignment3.pdf.**

**TASK:**

Implement stack using ArrayList



Program should be menu driven. The program should continue to run until the exit option is selected. Handel all corner cases.

|  |
| --- |
| **FUNCTION.H:**  #include <iostream> using namespace std;  class stack { public:  int top; //maintaining top as capacity of the stack  int size;  int \*array; //an array pointer   stack(int s) {  top = 0;  size = s;  array = new int[size]; //allocating the array of s size taken by user   }   bool empty() {  if (top == 0) {  return true; //function to check that if the top is zero then the stack is empty, and return true else false   } else {  return false;  }  }   bool full() {  if (top == size) {  return true; //checking that if the top is equal to the size that means there is no room left to add more elements  } else {  return false;  }  }   void showFront() {  if (empty()) {  throw underflow\_error("STACK IS EMPTY."); //checking if the array is empty otherwise throw an underflow error  } else {  cout << "THE FRONT VALUE OF STACK = ";  cout << array[top - 1]; //for showing the top element we have to print the top-1 element because the top is incremented everytime after adding 1 element in the stack which means it will be 1 place ahead to the place where the value is present   }  }   void push(int value) {  if (full()) {  throw overflow\_error("STACK IS FULL."); //checking if the array is full otherwise throw an overflow error   }  array[top] = value; //putting the value to the index where top is present   ++top; //then incrementing the top to move it forward   }   void pop() {  if (empty()) {  throw underflow\_error("STACK IS EMPTY."); //checking if the array is empty otherwise throw an underflow error  } else {  cout<<"THE POPPED ELEMENT = "<<array[top-1]; //the popped element is top-1 because top is incremented everytime in the push function   --top; //decrementing top to delete that element  }  }   void display() {  if (empty()) {  cout << " ";  }if(top!=0){  cout << "\nSTACK ELEMENTS ARE :";  for (int i = 0; i < top; i++)  cout << array[i] << " "; //displaying the stack  cout << endl;  }   }  ~stack() {  delete[]array; //destructor is called   } };  **MAIN.CPP:**  // // Created by Lenovo on 10/9/2022. // #include <iostream> #include "Functions.h"  using namespace std;  int main() {  int s;  int opt1;  int opt; //declaring  cout << "ENTER SIZE OF STACK =";  cin >> s; //taking the size and passing in constructor  stack T (s);  do {  cout << "\nCHOOSE OPTIONS." << endl; //showing options  cout << "1.PUSH ELEMENT." << endl;  cout << "2.SHOW FRONT." << endl;  cout << "3.POP ELEMENT." << endl;  cout << "4.EXIT." << endl;  cin >> opt;  if (opt == 1) {  int value;  cout << "ENTER THE VALUE TO ENTER IN STACK = ";  cin >> value;  T.push(value); //taking value and passing in the function  do {  cout << "YOU WANT TO PUSH AGAIN?" << "\nENTER 1 FOR YES AND 0 FOR NO" << endl;  cin >> opt1;  if (opt1 == 1) {  if (T.full()) { //checking if the array is full or not  cout<<"STACK IS FULL."<<endl;  break;  } else {  cout << "ENTER THE VALUE TO ENTER IN STACK = ";  cin >> value;  T.push(value); //if user enter 1 then calling the function again  T.display();  }  }  if (opt1 == 0) {  cout << "YOU CHOOSE NO." << endl;  break;  }  if (opt1 != 1 && opt1 != 0) {  cout << "YOU HAVE ENTER AN INVALID NUMBER." << endl;  break;  }  } while (opt1 >= 0 && opt1 <= 1);  T.display();  }  if (opt == 2) {  T.showFront();  }  if (opt == 3) {  if(T.empty()){ //checking if the array is empty  cout<<"STACK IS EMPTY."<<endl;  }  else{  T.pop();  T.display(); //calling  }  }  if(opt==4){  cout<<"YOU CHOOSE TO EXIT."<<endl;  exit(3); //exiting  }  }while (opt>=1 && opt<=4); //do while condition to run the menu until the user enter exit option.  return 0; }  **The explanation of code is done in the commenting.**  **Output:**    **Text  Description automatically generated** |

**TASK: 2**

**Write down the time complexity(in Big O) of the following Codes**

**int f1( int n) {**

**Int k=0;**

**for ( int j =0; j ∗ j<=n ; j++)**

**K++;**

**return k ;**

**}**

**int main ( ) {**

**int sum =0 ;**

**int n ;**

**cin>>n ;**

**int terminator = f1 (n) ;**

**for ( int i =1; i<=terminator ; i+=1)**

**for ( int j =1; j<=i ; j++)**

**sum++;**

**cout<<sum<<endl ;**

**}**

|  |
| --- |
| **COST REPITITION TOTAL**  **1 1 1**  **1 1 1**  **1 sqrt(n) sqrt(n)**  **1 sqrt(n) sqrt(n)**  **1 1 1**  **1 1 1**  **1 sqrt(n) sqrt(n)**  **1 sqrt(n) sqrt(n)**  **1 n n**  **Time complexity = O(sqrt(n).** |