

**National University of Computer and Emerging**

**Sciences**

**Chiniot-Faisalabad Campus BS (Artificial Intelligence)**

|  |  |
| --- | --- |
| **Name** | **M Abdul Hanan** |
| **Reg.NO.** | **22F-3104** |
| **Section** | **BS(Ai)-3A** |
| **Course** | **Data Structures** |
| **Department** | **CS Department** |
| **Lab** | **#5** |

**Task no 1:**

#include <iostream>

using namespace std;

//in this program in which i create a stack which uses LIFO method to perform different operations such as pop to delete last elemnt from the last or push to insert at last

class stack {

private:

int top;

int rows;

int\* arr;

public:

stack() {//default constructor where we give default value

top = -1;

cout << "Enter the number of rows: ";

cin >> rows;

arr = new int[rows];

for (int i = 0; i < rows; i++) {

arr[i] = 0;

}

}

bool isEmpty() {//check stack is empty or not

return (top == -1);

}

bool isFull() {//check stack is full or notr

return (top == rows - 1);

}

void push(int val) {//push function to insert at last in the stack

if (isFull()) {

cout << "Stack is full." << endl;

}

else {

top++;

arr[top] = val;

}

}

int pop() {//delete element at the last of the stack

if (isEmpty()) {

cout << "Stack is empty." << endl;

return 0;

}

else {

int popValue = arr[top];

arr[top] = 0;

top--;

return popValue;

}

}

void display() {//display the stack

cout << "All values in the stack are:" << endl;

for (int i = top; i >= 0; i--) {

cout << arr[i] << " ";

}

cout << endl;

}

~stack() {

delete[] arr;

}

};

int main() {

stack s1;

int num = 0;

do {

cout << "Enter your choice (1: Push, 2: Pop, 3: Check if empty, 4: Check if full,5:display 0: Exit): ";//menu driven

cin >> num;

if (num == 1) {//choices given to choose from menu

cout << "Enter number you want to push: ";

int num;

cin >> num;

s1.push(num);

}

else if (num == 2) {

s1.pop();

}

else if (num == 3) {

if (s1.isEmpty()) {

cout << "Stack is empty :" << endl;

}

else {

cout << "Stack is not empty :" << endl;

}

}

else if (num == 4) {

if (s1.isFull()) {

cout << "Stack is full :" << endl;

}

else {

cout << "Stack is not full :" << endl;

}}

else if (num == 5) {

cout << "print the Stack :" << endl;

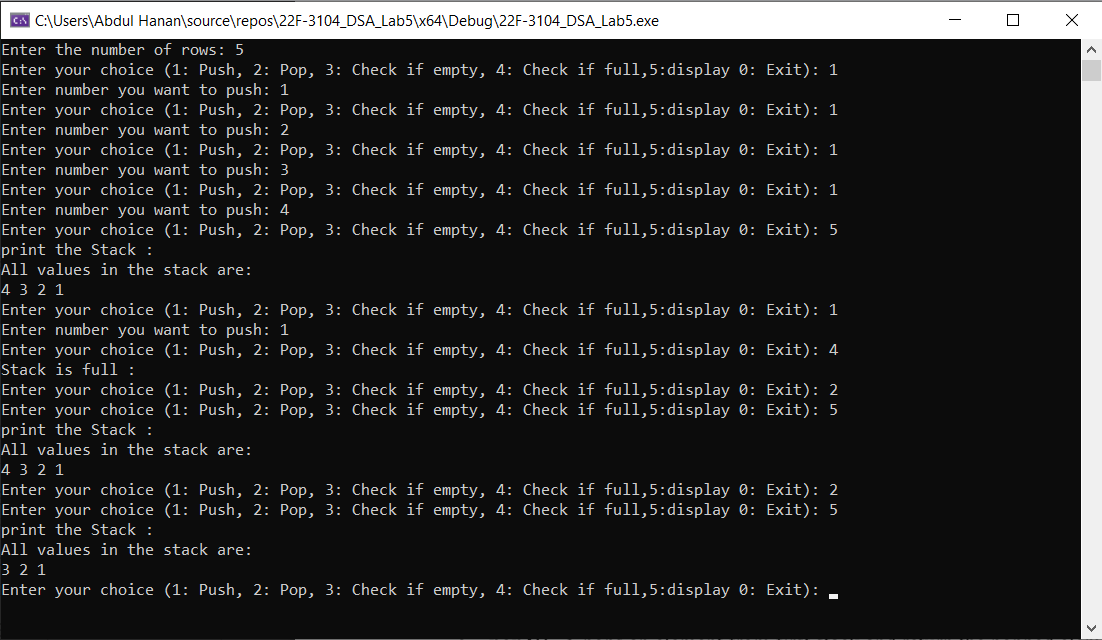
s1.display();

}

} while (num != 0);

return 0;

}

****

**Task no 2:**

#include <iostream>

using namespace std;

class TwoStacks {

private:

int\* arr;

int size;

int top1;

int top2;

public:

TwoStacks(int rows) {

size = rows;

arr = new int[rows];

top1 = -1; // Initialize top pointer for the first stack

top2 = size; // Initialize top pointer for the second stack

}

// Push an element to the first stack

void push1(int val) {

if (top1 < top2 - 1) { // Check if there is space for the first stack

++top1;

arr[top1] = val;

}

else {

cout << "Stack 1 is full." << endl;

}

}

// Push an element to the second stack

void push2(int val) {

if (top1 < top2 - 1) { // Check if there is space for the second stack

top2--;

arr[top2] = val;

}

else {

cout << "Stack 2 is full." << endl;

}

}

// Pop an element from the first stack and return it

int pop1() {

if (top1 >= 0) { // Check if the first stack is not empty

return arr[top1--];

}

else {

cout << "Stack 1 is empty." << endl;

return -1; // Return a sentinel value that empty stack

}

}

// Pop an element from the second stack and return it

int pop2() {

if (top2 < size) { // Check if the second stack is not empty

return arr[top2++];

}

else {

cout << "Stack 2 is empty." << endl;

return -1; // Return a sentinel value to indicate an empty stack

}

}

void display() {

// Print stack 1:

cout << "Stack 1 is: ";

for (int i = 0; i <= top1; i++) {

cout << arr[i] << " ";

}

cout << endl;

// Print stack 2:

cout << "Stack 2 is: ";

for (int i = top2; i < size; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

~TwoStacks() {

delete[] arr;

}

};

int main() {

int size;

cout << "Enter the size of the array: ";

cin >> size;

TwoStacks ts(size);

int choice;

do {

cout << "Enter your choice (1: Push, 2: Pop, 3: Display, 0: Exit): ";

cin >> choice;

if (choice == 1) {

cout << "Enter which stack you want to push value (1 for stack 1, 2 for stack 2): ";

int stackNum;

cin >> stackNum;

if (stackNum == 1) {

cout << "Enter number you want to push in stack 1: ";

int num1;

cin >> num1;

ts.push1(num1);

}

else if (stackNum == 2) {

cout << "Enter number you want to push in stack 2: ";

int num2;

cin >> num2;

ts.push2(num2);

}

}

else if (choice == 2) {

cout << "Enter which stack you want to pop from (1 for stack 1, 2 for stack 2): ";

int stackNum;

cin >> stackNum;

if (stackNum == 1) {

int poppedValue = ts.pop1();

if (poppedValue != -1) {

cout << "Popped value from stack 1: " << poppedValue << endl;

}

}

else if (stackNum == 2) {

int poppedValue = ts.pop2();

if (poppedValue != -1) {

cout << "Popped value from stack 2: " << poppedValue << endl;

}

}

}

else if (choice == 3) {

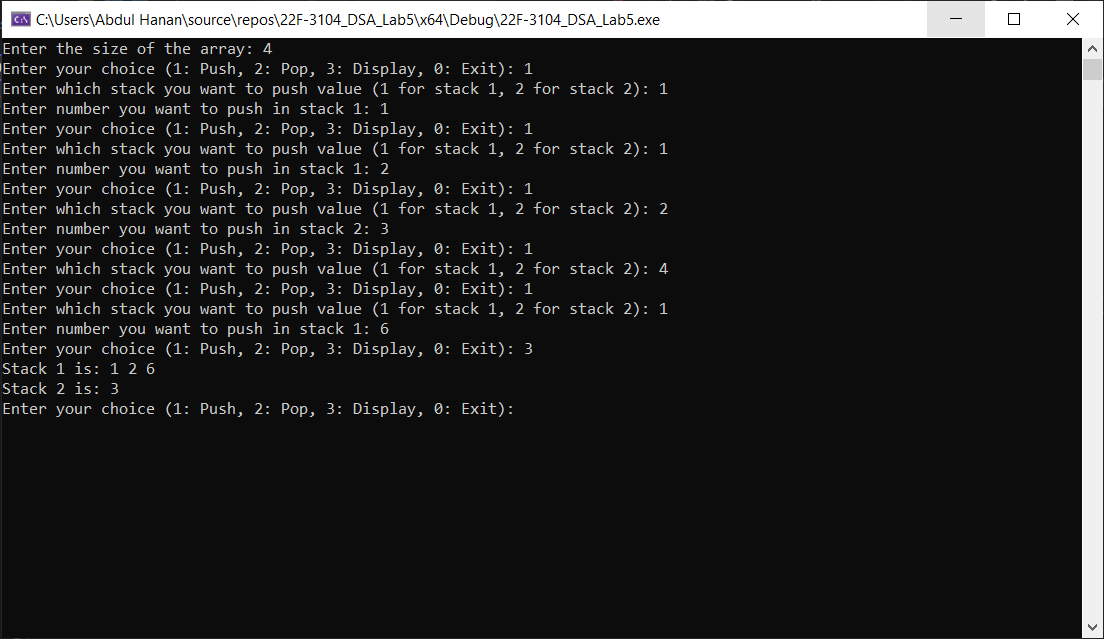
ts.display();

}

} while (choice != 0);

return 0;

}

****

**Task no 3:**

#include <iostream>

//in this program i create a stack using link list perfroming the same operation as stack but by using singly link list

using namespace std;

class StackNode {//create Nodestack

public:

int data;

StackNode\* next;

StackNode(int val) {

data = val;

next = nullptr;

}

};

class Stack {

public:

StackNode\* top;

Stack() {

top = nullptr;

}

void Push(int val) {//push value in the link list at the tail

StackNode\* newNode = new StackNode(val);

newNode->next = top;

top = newNode;

}

void Pop() {//pop value in the link list at the last of the link list

if (IsEmpty()) {

cout << "Stack is empty. Cannot pop" << endl;

return;

}

StackNode\* temp = top;

top = top->next;

delete temp;

}

bool IsEmpty() {//check list is empty or not

return top == nullptr;

}

void Display\_stack() {//display the stack

if (IsEmpty()) {

cout << "Stack is empty." << endl;

return;

}

StackNode\* current = top;

cout << "Stack: ";

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

void Reverse\_stack() {//reverse the linklist as keeping the functionality of stack

StackNode\* prev = nullptr;

StackNode\* current = top;

StackNode\* next = nullptr;

while (current != nullptr) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

top = prev;

}

};

int main() {

Stack stack;

int choice;

int num;

do {//menu driven of the program

cout << "Press 1: Push, 2: Pop, 3: IsEmpty, 4: Reverse\_Stack, 5: Display\_Stack, 0: Exit: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter number to push in the stack: ";

cin >> num;

stack.Push(num);

break;

case 2:

stack.Pop();

break;

case 3:

if (stack.IsEmpty()) {

cout << "Stack is empty." << endl;

}

else {

cout << "Stack is not empty." << endl;

}

break;

case 4:

stack.Reverse\_stack();

break;

case 5:

stack.Display\_stack();

break;

case 0:

cout << "Exiting the program." << endl;

break;

default:

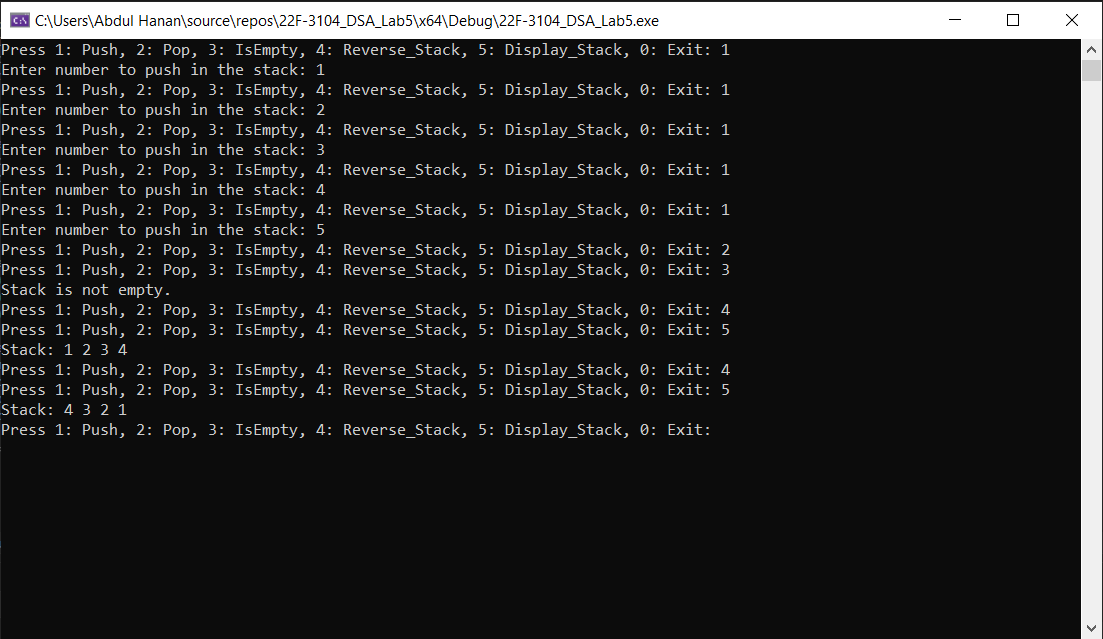
cout << "Invalid choice. Try again." << endl;

}

} while (choice != 0);

return 0;

}

****

**Task no 4:**

#include <iostream>

using namespace std;

//in this i create a linklist using stack method to calculate the grocery shop

class StackNode {

public:

string item;

double price;

StackNode\* next;

StackNode(const string& itemName, double itemPrice) {//parameterized constructor to intialize the value

item = itemName;

price = itemPrice;

next = nullptr;

}

};

class Stack {

public:

StackNode\* top;

Stack() {

top = nullptr;

}

void Push(string& itemName, double itemPrice) {//function to push element in the

StackNode\* newNode = new StackNode(itemName, itemPrice);

newNode->next = top;

top = newNode;

}

void Pop() {//pop the element in ths stack

if (IsEmpty()) {

cout << "Shopping list is empty" << endl;

return;

}

StackNode\* temp = top;

top = top->next;

delete temp;

}

bool IsEmpty() {//check it is empty or not

return top == nullptr;

}

void Peek() {//check next item which is remaining in this list

if (IsEmpty()) {

cout << "Shopping list is empty." << endl;

}

else {

cout << "Next item to purchase: " << top->item << " " << top->price << endl;

}

}

void ListItems() {//display the list of elements in the stack

if (IsEmpty()) {

cout << "Shopping list is empty." << endl;

return;

}

cout << "Shopping list:" << endl;

StackNode\* current = top;

while (current != nullptr) {

cout << current->item << " " << current->price << " " << endl;

current = current->next;

}

}

double CalculateBill() {//calcualte the total bill

double totalBill = 0.0;

StackNode\* current = top;

while (current != nullptr) {

totalBill += current->price;

current = current->next;

}

return totalBill;

}

};

int main() {

Stack shoppingList;

int choice;

string item;

double price;//menu driven program

do {

cout << "1: insert item, 2: Pop item, 3: check next item, 4: Is Empty, 5:Print List Items, 6: Calculate Bill, 0: Exit: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter item name: ";

cin >> item;

cout << "Enter item price: ";

cin >> price;

shoppingList.Push(item, price);

break;

case 2:

shoppingList.Pop();

break;

case 3:

shoppingList.Peek();

break;

case 4:

if (shoppingList.IsEmpty()) {

cout << "Shopping list is empty." << endl;

}

else {

cout << "Shopping list is not empty." << endl;

}

break;

case 5:

shoppingList.ListItems();

break;

case 6:

cout << "Total bill: " << shoppingList.CalculateBill() << endl;

break;

case 0:

cout << "Exiting the shopping list app." << endl;

break;

default:

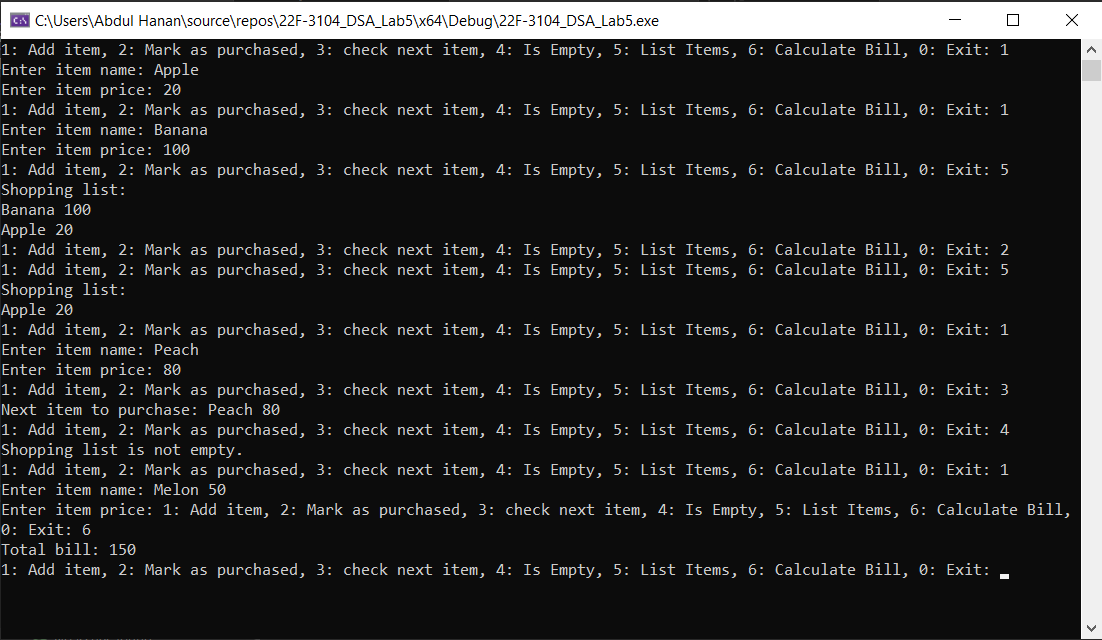
cout << "Invalid choice Try again." << endl;

}

} while (choice != 0);

return 0;

}

****

**Task no 5:**

#include<iostream>

using namespace std;

struct StackNode {

char num;

StackNode\* next;

StackNode(char t) {

num = t;

next = NULL;

}

};

class stacklinkedlst {

StackNode\* top;

string data;

public:

stacklinkedlst() {

top = NULL;

}

// Create a new stack node

StackNode\* createStackNode(char t) {

StackNode\* nn = new StackNode(t);

return nn;

}

// Check if the stack is empty

bool isempty() {

return (top == NULL);

}

// Insert characters into the stack and process parentheses

void insert(char ch) {

if (ch == '(') {

push(ch);

}

else if (ch == ')') {

while (!isempty() && top->num != '(') {

char popped = pop();

data = data + popped;

}

pop();

}

}

// Process any remaining characters in the stack

void remaining() {

while (!isempty()) {

char ch = pop();

data = data + ch;

}

}

// Push a character onto the stack

void push(char ch) {

StackNode\* newStackNode = new StackNode(ch);

if (top == nullptr) {

top = newStackNode;

}

else {

newStackNode->next = top;

top = newStackNode;

}

}

// Pop a character from the stack

char pop() {

char ch;

StackNode\* tempptr = top;

if (tempptr != NULL) {

if (tempptr->next != NULL) {

top = tempptr->next;

}

else {

top = NULL;

}

ch = tempptr->num;

delete tempptr;

}

else {

cout << "list is empty";

ch = '\0';

}

return ch;

}

};

int main() {

stacklinkedlst s1;

string expression;

int num = 0;

do {

cout << "Enter any number or press (0) to exit :" << endl;

cin >> num;

cout << "Enter an expression: ";

cin >> expression;

bool isBalanced = true;

// Loop through each character in the expression

for (char ch : expression) {

if (ch == '(' || ch == '[' || ch == '{') {

s1.push(ch);

}

else if (ch == ')' || ch == ']' || ch == '}') {

if (s1.isempty()) {

isBalanced = false;

break;

}

char popped = s1.pop();

if ((ch == ')' && popped != '(') || (ch == ']' && popped != '[') || (ch == '}' && popped != '{')) {

isBalanced = false;

break;

}

}

}

// Check if the stack is empty after processing the expression

if (!s1.isempty()) {

isBalanced = false;

}

// Output whether the expression is balanced or not

if (isBalanced) {

cout << "The expression is balanced " << endl;

}

else {

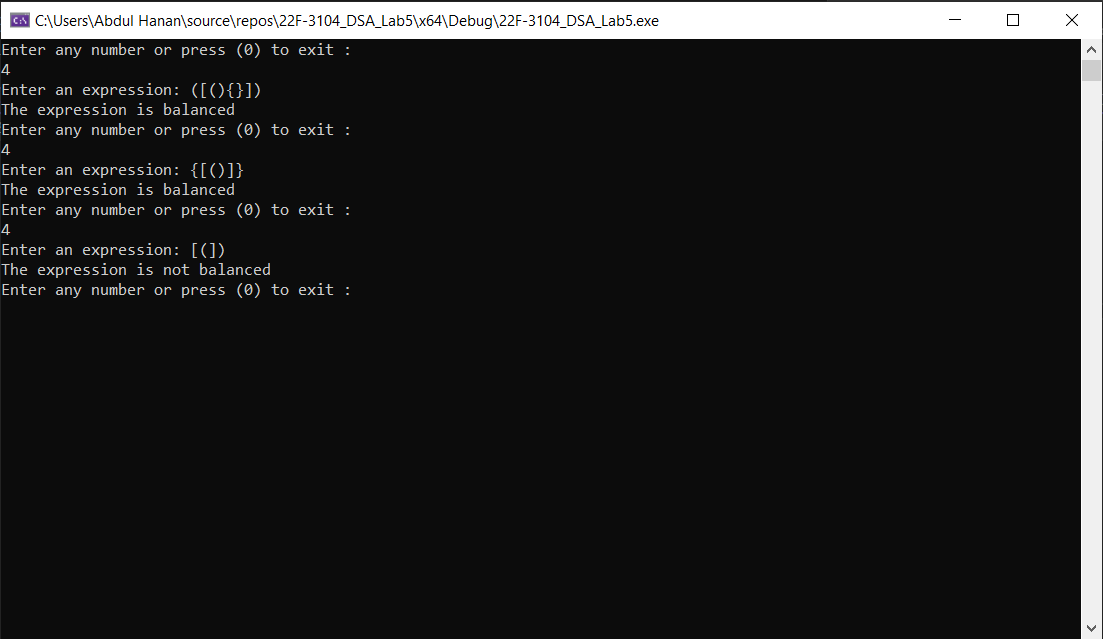
cout << "The expression is not balanced " << endl;

}

} while (num != 0);

return 0;

}

****