TEHNIČKA ŠKOLA RUĐERA BOŠKOVIĆA	
Naziv modula:	Uvod u programiranje
Skup ishoda učenja:	SUI2: Spremnički tipovi podataka
Naziv laboratorijske vježbe (UTR):	**LV21: Spremnik tipa stog (stack, LIFO) (UTR)**
Ishodi učenja:	– prepoznati problemske situacije za primjenu spremnika tipa stog – pravilno kodirati spremnik tipa stog – pravilno primjenjivati osnovne operacije na stogu – pravilno kodirati osnovne algoritme na stogu – primjenjivati gotove funkcije/metode na stogu – uspješno prepoznavati i ispravljati pogreške u oblikovanju algoritama primjenjivih na stogu
Ime i prezime učenika/datum	

UPUTE

Ako je prilikom rješavanja zadataka korišten generativni AI treba navesti i pitanja (promptove). Nerazumijevanje prezentiranog (AI generiranog) programskog koda će utjecati na ocjenu, tako da je nužno sav kod koji se ubacuje i prezentira kao odgovor pregledati, proučiti, razmisliti o njemu i shvatiti, te po potrebi ispraviti i znati ispraviti, dopuniti i/ili komentirati. Za nejasnoće i/ili dodatne upute obratiti se nastavniku. Vježbu za ocjenjivanje može se predati najkasnije do 10 minuta prije kraja sata.

PRIPREMA:

1. Objasnite osnovne funkcije i metode na stogu u C++.

I. PROBLEMSKI ZADACI (s rješenjima)

Proučite i usporedite implementacije stoga u nastavku

1. PROBLEM - ZAGRADE

```
/* 1. primjer
C++ Program koji provjerava uravnoteženost zagrada u izrazu uz pomoć
stoga. Za dani izraz u obliku stringa koji sadrži okrugle otvorene i
zatvorene zagrade-znakove - (), vitičaste - {} i kvadratne - [],
treba provjeriti jesu li simboli uravnoteženi.
*/

#include<iostream>
#include<stack>
#include<string>
using namespace std;

// Funkcija koja provjerava jesu li dva znaka otvoreni i zatvoreni
// istog tipa.

bool JesuLiPar(char opening,char closing)
{
	if(opening == '(' && closing == ')') return true;
	else if(opening == '{' && closing == '}') return true;
	else if(opening == '[' && closing == ']') return true;
	return false;
}
bool JesuLiZagradeUravnotezene(string izrazTest)
{
	stack<char>  S;
	for(int i =0;i<izrazTest.length();i++)
	{
		if(izrazTest[i] == '(' || izrazTest[i] == '{' ||
izrazTest[i] == '[')
			S.push(izrazTest[i]);
		else if izrazTest [i] == ')' || izrazTest [i] == '}' ||
izrazTest [i] == ']')
		{
			if(S.empty() || !JesuLiPar(S.top(),izrazTest [i]))
				return false;
			else
				S.pop();
		}
	}
	return S.empty() ? true:false;
}

int main()
{
	/*kod za ispitivanje funkcije JesuLiZagradeUravnotezene*/
	string izraz;
	cout<<"Unesi izraz:  "; cin>>izraz;
	if(JesuLiZagradeUravnotezene(izraz))
		cout<<"Uravnotezene\n";
	else
		cout<<"Nisu uravnotezene\n";}
```

```
//2. primjer - jednostavni program koji provjerava nedostaju li
//zagrade

#include <iostream>
#include <stack>
#include <string>

using namespace std;

// vraća jesu li u ulaznom nizu zagrade uravnotežene

bool provjeraZagrada(string izraz) {
    stack<string> s;
    bool balanced = true;
    int index = 0;
    int str_len = izraz.length();

    while (index < str_len && balanced) {
           string symbol;
           symbol = izraz[index];
           if (symbol == "(") {
               s.push(symbol); //stavlja otvorenu zagradu na stog
           } else {
               if (s.empty()) {
//ako nema otvorenih zagrada na stogu upravo nađena zatvorena
//zagrada čini string neuravnoteženim.
                balanced = false;
            } else {
//ako je otvorena zagrada na stogu,upravo nađena zatvorena zagrada
//ima par.
              s.pop();
            }
    }
    index = index + 1;
    }

    if (balanced && s.empty()) {
//ako je string uravnotežen i nema preostalih otvorenih zagrada.
        return true;
    } else {
        return false;
    }
}

int main() {
    cout << provjeraZagrada("((()))") << endl;
    cout << provjeraZagrada("(()") << endl;
}
```

```
//3. primjer - program koji vraća je li ulazni string uravnotežen

#include <iostream>
#include <string>
#include <stack>

using namespace std;

//vraća istinu ako je simbol u skupu simbola
bool inString(string symbol, string symbols){
    return symbols.find(symbol) != string::npos;}

//vraća istinu ako je simbol u skupu "open" par sa simbolom u
"close"
bool matches(string open, string close){
    string opens = "({[";
    string closers = ")]}";
    bool val = inString(open, opens) == inString(close, closers);
    return val;}

// Prolazi kroz string stavljajući simbol iz skupa otvorenih
// i uzimajući odgovarajući zatvoreni simbol ako se pojavio
bool parChecker(string symbolString){
    stack<string> s;
    bool balanced = true;
    int index = 0;
    int symbolLength = symbolString.length();

    while(index < symbolLength && balanced){
        string symbol;
        symbol = symbolString[index];
        string opens = "([{";
        string closes = "}])";
        if (inString(symbol, opens)){
            s.push(symbol);
        } else if(inString(symbol, closes)){
            if (s.empty()){
                balanced = false;
            } else {
                string top = s.top();
                s.pop();
                if (!matches(top, symbol)){
                    balanced = false;
                    break;  } }}
        index = index + 1; }
    if(balanced && s.empty()){
        return true;
    } else {
        return false; }}

int main() {
    cout << parChecker("{}") << endl;
    cout << parChecker("[{()}]") << endl;
    return 0;}
```

2. PROBLEM - PRETVORBE

```
//4. primjer: pretvara dani decimalni broj u binarni pomoću stoga.

#include <iostream>
#include <stack>
#include <string>

using namespace std;

string divideBy2(int decNumber) {
  //provodi proces konverzije.
  stack<int> remstack;

  while (decNumber > 0) {
          //dobija ostatak dijeljenja sa 2
          //i stavlja ostatak na stog.
          int rem = decNumber % 2;
          remstack.push(rem);
          decNumber = decNumber / 2;
  }

  string binString = "";
  while (!remstack.empty()) {
          //dodaje brojeve-ostatke sa stoga u string.
          binString.append(to_string(remstack.top()));
          remstack.pop();
  }

  return binString;
}

int main() {
  cout << divideBy2(42) << endl;

  return 0;
}
```

Napravite programsko rješenje za pretvorbu decimalnog broja u oktalni.

3. PROBLEM

Napravite (i/ili proučite postojeće) programsko rješenje za primjere sa stranice. Proučite ilustracije/animacije algoritamskog pristupa.