**Lab 3 – Stacks**

Screenshot of 5 test cases for palindrome checker:A screenshot of a computer program

AI-generated content may be incorrect.

Screenshot of 5 test cases for symbol checker:A screenshot of a computer program

AI-generated content may be incorrect.

Screenshot of 5 test cases for Number Base Converter:A screenshot of a computer program

AI-generated content may be incorrect.

Screenshot of 5 test cases for In2PostFix Translator:A screen shot of a computer program

AI-generated content may be incorrect.

**Code Appendix**

This first snippet of code contains both the palindrome and symbol checker functions:

1. #include "charStack.h"

2. #include <string>

3. #include <iostream>

4. using namespace std;

5.

6. // function to check if the given string is a palindrome

7.

8. bool isPalindrome(const string& str) {

9. CharStack stack = ' ';

10. string res;

11.

12. //Decided I wanted to remove spaces from the string since your instructions say

13. //check if the \*string\* is a palindrome, not just a single word.

14. for (char c : str) {

15. if (c != ' ') {

16. res += c;

17. }

18. }

19.

20. int len = res.length();

21.

22. for (int i = 0; i < len; i++) {

23. stack.push(res[i]);

24. }

25.

26. for (int i = 0; i < len; i++) {

27. if (res[i] != stack.pop()) {

28. return false;

29. }

30. }

31. return true;

32. }

33.

34.

35. int main() {

36. string str;

37.

38. cout << "Enter a string: ";

39. getline(cin, str);

40.

41. if (isPalindrome(str)) {

42. cout << "The string is a palindrome." << endl;

43. } else {

44. cout << "The string is not a palindrome." << endl;

45. }

46. return 0;

47. }

48.

49. // check if opening and closing symbols match

50. bool matches(char openSymb, char closeSymb) {

51. string opens = "([{";

52. string closers = ")]}";

53.

54. size\_t openPos = opens.find(openSymb);

55. size\_t closePos = closers.find(closeSymb);

56.

57. return (openPos != s:tring::npos && closePos != string::npos && openPos == closePos);

58. }

59.

60. // check if symbols in a string are balanced

61. bool symChecker(const string& symbolString) {

62. CharStack s('0'); // Initialize with a character (seems to be used for capacity)

63. bool balanced = true;

64. size\_t index = 0;

65.

66. while (index < symbolString.length() && balanced) {

67. char symbol = symbolString[index];

68.

69. if (symbol == '(' || symbol == '[' || symbol == '{') {

70. // an opener

71. s.push(symbol);

72. } else if (symbol == ')' || symbol == ']' || symbol == '}') {

73. // a closer

74. if (s.isEmpty()) {

75. balanced = false;

76. } else {

77. char top = s.pop();

78. if (!matches(top, symbol)) {

79. balanced = false;

80. }

81. }

82. }

83.

84. index++;

85. }

86.

87. return balanced && s.isEmpty();

88. }

89.

90. // function for generic symbol checker

91. int main() {

92. string symb;

93.

94. cout << "Enter a string of symbols to be checked: ";

95. getline(cin, symb);

96. if (symChecker(symb)) {

97. cout << "The symbols are balanced." << endl;

98. } else{

99. cout << "The symbols are NOT balanced." << endl;

100. }

101. return 0;

102. }

103.

Binary Converter code:

1. #include <iostream>

2. #include <string>

3. #include "intStack.h"

4.

5. using namespace std;

6.

7. string dec2Bin(int decNumber) {

8. int sz = 100;

9. IntStack remstack(sz);

10.

11. while (decNumber > 0) {

12. int rem = decNumber % 2;

13. remstack.push(rem);

14. decNumber = decNumber / 2;

15. }

16.

17. string binString;

18. while (!remstack.isEmpty()) {

19. binString += to\_string(remstack.pop());

20. }

21.

22. return binString;

23. }

24.

25. string dec2Base(int decNumber, int base) {

26. const string digits = "0123456789ABCDEF";

27. int sz = 100;

28. IntStack remstack(sz);

29.

30. while (decNumber > 0) {

31. int rem = decNumber % base;

32. remstack.push(rem);

33. decNumber = decNumber / base;

34. }

35.

36. string newString;

37. while (!remstack.isEmpty()) {

38. newString += digits[remstack.pop()];

39. }

40.

41. // I struggled with this function and, in particular, the handling of this string.

42. // I kept getting bad output and had to get some help to remember this step...

43. reverse(newString.begin(), newString.end());

44.

45. return newString;

46. }

47.

48. int main(){

49. int decNumber;

50. int base;

51.

52. cout << "Enter a decimal number: ";

53. cin >> decNumber;

54.

55. cout << "Binary representation: " << dec2Bin(decNumber) << endl;

56.

57. cout << "Enter a base (2-16): ";

58. cin >> base;

59.

60. if (base >= 2 && base <=16) {

61. cout << "Base " << base << " representation: " << dec2Base(decNumber, base) << endl;

62. } else {

63. cout << "Please enter a valid base, nerd." << endl;

64. }

65.

66. return 0;

67. }

68.

In2Post Code:

1. #include "charStack.h"

2. #include <iostream>

3. #include <string>

4. #include <vector>

5.

6. using namespace std;

7.

8. // I struggled with changing this over to c++ so I got some help with it. I understand it, but I had

9. // a hard time with this one.

10. string in2post(const string& infixexpr) {

11. int prec[128] = {0};

12. prec['\*'] = 3;

13. prec['/'] = 3;

14. prec['+'] = 2;

15. prec['-'] = 2;

16. prec['('] = 1;

17.

18. CharStack opStack(10); // I wasn't sure what to initialize this with, so I just picked 10

19. vector<string> postfixList;

20. string token;

21. string result;

22.

23. for (char ch : infixexpr) {

24. if (isalpha(ch)) {

25. result += ch;

26. result += ' ';

27. } else if (ch == '(') {

28. opStack.push(ch);

29. } else if (ch == ')') {

30. char topToken = opStack.pop();

31. while (topToken != '(') {

32. result += topToken;

33. result += ' ';

34. topToken = opStack.pop();

35. }

36. } else if (ch == ' ') {

37. continue;

38. } else {

39. while (!opStack.isEmpty() && prec[opStack.peek()] >= prec[ch]) {

40. result += opStack.pop();

41. result += ' ';

42. }

43. opStack.push(ch);

44. }

45. }

46.

47. while (!opStack.isEmpty()) {

48. result += opStack.pop();

49. result += ' ';

50. }

51.

52. if (!result.empty() && result.back() == ' ') {

53. result.pop\_back(); // Remove the trailing space

54. }

55.

56. return result;

57. }

58.

59. int main() {

60. string infixexpr;

61. cout << "Enter an infix expression: ";

62. getline(cin, infixexpr);

63.

64. string postfix = in2post(infixexpr);

65. cout << "Postfix expression: " << postfix << endl;

66.

67. return 0;

68. }

69.

PF Interpreter code:

1. include "intStack.h"

2. #include <iostream>

3. #include <string>

4.

5. using namespace std;

6.

7. int evaluatePostfix(const string& postfix) {

8. IntStack operandStack(10); // Initial size, constructor will add 100

9. string token;

10.

11. for (size\_t i = 0; i < postfix.length(); ++i) {

12. char ch = postfix[i];

13. if (isdigit(ch)) {

14. operandStack.push(ch - '0');

15. } else if (ch == ' ') {

16. continue;

17. } else {

18. int operand2 = operandStack.pop();

19. int operand1 = operandStack.pop();

20. int result;

21.

22. switch (ch) {

23. case '+':

24. result = operand1 + operand2;

25. break;

26. case '-':

27. result = operand1 - operand2;

28. break;

29. case '\*':

30. result = operand1 \* operand2;

31. break;

32. case '/':

33. result = operand1 / operand2;

34. break;

35. default:

36. cout << "Error: Unknown operator " << ch << endl;

37. return -1;

38. }

39.

40. operandStack.push(result);

41. }

42. }

43.

44. return operandStack.pop();

45. }

46.

47. int main() {

48. string postfix;

49. cout << "Enter a postfix expression: ";

50. getline(cin, postfix);

51.

52. int result = evaluatePostfix(postfix);

53. cout << "Result: " << result << endl;

54.

55. return 0;

56. }

57.

intStack.h code:

1. #include "intStack.h"

2. #include "charStack.h"

3. #include <iostream>

4. #include <string>

5. #include <vector>

6.

7. class IntStack {

8. public:

9. intStack(int sz); // Constructor

10. bool isEmpty();

11. void push(int item);

12. int peek();

13. int pop();

14. int size();

15. private:

16. std::vector<int> vec;

17. };

18.

19. inStack::intStack(int sz) {

20. vec.reserve(sz + 100);

21. }

22.

23. bool IntStack::isEmpty() {

24. return vec.size() == 0;

25. }

26.

27. void IntStack::push(int item) {

28. vec.push\_back(item);

29. }

30.

31. int IntStack::peek() {

32. if (this->isEmpty()) {

33. cout << "Error: Peeking an empty intStack: NULL returned." << endl;

34. return NULL;

35. }

36. else {

37. return vec.back();

38. }

39. }

40.

41. int IntStack::size() {

42. return vec.size();

43. }

44.

45. int IntStack::pop() {

46. int res;

47. if (this->isEmpty()) {

48. cout << "Error: Popping an empty intStack: NULL returned." << endl;

49. return NULL;

50. }

51. else {

52. res = vec.back();

53. vec.pop\_back();

54. return res;

55. }

56. }

57.

charStack.h code:

1. #include "intStack.h"

2. #include "charStack.h"

3. #include <iostream>

4. #include <string>

5. #include <vector>

6.

7. class CharStack {

8. public:

9. charStack(char c); // Constructor

10. void push(char item);

11. bool isEmpty();

12. char pop();

13. char peek();

14. char size();

15. private:

16. std::vector<char> vec;

17. };

18.

19. charStack::charStack(char c) {

20. vec.reserve(c + 100);

21. }

22.

23. void CharStack::push(char item) {

24. vec.push\_back(item);

25. }

26.

27. bool CharStack::isEmpty() {

28. return vec.size() == 0;

29. }

30.

31. char CharStack::pop() {

32. char res;

33. if (this->isEmpty()) {

34. cout:: << "Error: Popping an empty charStack: NULL returned." << endl;

35. return NULL;

36. }

37. else {

38. return vec.back();

39. }

40.

41. char CharStack::peek() {

42. if (this->isEmpty()) {

43. cout << "Error: Peeking an empty charStack: NULL returned." << endl;

44. }

45. else {

46. return vec.back();

47. }

48. }

49.

50. char CharStack::size() {

51. return vec.size();

52. }

53.