Assignment - 3

**Q1.** Pusedocode to check a palindrome string with stack.

1. define Max 100 and initialize stack[MAX], top=-1 and front=0.

2. push(char a)

top++

Stack[top] = a.

3. pop()

top--

4. main()

i) Initialize string s[100];

ii) get a string from user

iii) For I = 0 to null

Char c = s[i]

Push(c)

iv) For I = 0 to [length of string / 2]

a) If stack[top] = stack[front]

b) Pop()

c) front++

d) else

e) print “string is not palindrome.”

f) break

v) if (strlen(s)/2) == front

print “string is palindrome.”

vi) Return 0.

**Q2.** Psuedocode to convert infix expression into prefix.

i) define Max 100 and initialize stack[MAX], top =-1.

ii) isEmpty()

If top < 0

return -1

return 0;

iii) push(char x)

stack[++top] = x.

iv) pop()

if(!isEmpty())

return

stack[top--].

v) peek()

return stack[top].

vi) precedence(char x)

If x = '('

return 0

if x = '+' or x = '-'

return 1

if x = '\*' or x = '/'

return 2

vii) checkIfOperand(char ch)

return ( ch >= 'a and ch <= 'z' ) or ( ch >= 'A and ch <= 'Z' )

vii) getPostfix(char exp[])

int i, j

for i = 0, j = -1; exp[i]; ++i

if checkIfOperand(exp[i])

exp[++j] = exp[i]

else if exp[i] == '('

push(exp[i])

else if exp[i] = ')'

while !isEmpty() and peek(stack) != '('

exp[++j] = pop()

if !isEmpty() and peek() != '('

return -1

else

pop()

else

while !isEmpty() and precedence(exp[i] <= precedence(peek())

exp[++j] = pop()

push(exp[i])

while !isEmpty()

exp[++j] = pop()

exp[++j] = '\0'

ix) reverse(char exp[])

int size = strlen(exp)

int j = size, i=0

char temp[size]

temp[j--]='\0'

while(exp[i]!='\0')

temp[j] = exp[i]

j--

i++

strcpy(exp,temp).

x) brackets(char exp[])

int i = 0

while exp[i]!='\0'

if exp[i]=='('

exp[i]=')'

else if exp[i]==')'

exp[i]='('

i++.

xi) InfixtoPrefix(char exp[])

int size = strlen(exp)

reverse(exp)

brackets(exp)

getPostfix(exp)

reverse(exp)

xii) main()

char exp[100];

print "The infix is: ".

gets(exp)

InfixtoPrefix(exp)

Print "The prefix is: ".

Print exp

return 0

**Q3.** Convert the following expressions into prefix and postfix using stack:

1) a\*(b-c\*d)+e

2) a+((b-c)\*d)/e

|  |  |  |
| --- | --- | --- |
| Infix expression | Postfix expression | Prefix expression |
| a\*(b-c\*d)+e | abcd\*-\*e+ | +\*a-b\*cde |
| a+((b-c)\*d)/e | abc-d\*e/+ | +a/\*-bcde |

**Q4.** Psuedocode to evaluate a postfix expression.

1. define Max 100 and initialize stack[MAX] , top = -1.

2. push(int ele)

If top >= MAX-1

Print “stack overflow”.

else

Top = top + 1;

Stack[top] = ele

3. pop()

If top < 0

print “stack under flow”.

Return

else

Int item = stack[top]

top = top - 1

return item.

4.Evaluate(char exp)

Initialize A, B and ans.

For int i=0; exp[i] != ‘\0’; i++

char ch = exp[i]

if isdigit(ch)

push(ch-‘0’)

else if ch==’+’ or ch==’-‘ or ch==’\*’ or ch==’/’ or ch==’$’

B = pop()

A = pop()

switch (ch)

case '\*': ans = A \* B

break

case '/': ans = A / B

break

case '+': ans = A + B

break

case '-': ans = A - B

break

case '$': ans = pow(A,B)

break

push(ans);

print “Result of expression evaluation : pop()”.

5. main()

char exp[MAX]

print "Enter an Expression: ".

scanf("%s",&exp)

Evaluate(exp)

Return 0

**Q5.** Evaluate the following expressions using stack:

1) 34+86-\*

2) 222$$3\*2+2\*

|  |  |
| --- | --- |
| expression | Postfix evaluation |
| 1) 34+86-\* | 14 |
| 2) 222$$3\*2+2\* | 100 |

**Q6.** Psuedocode for Fibonacci series with recursion.

1. declare fiboinacci function.

2. main()

declare n.

print “Enter a number of terms:”

scan as n.

print n “th term:”.

call Fibonacci function.

return 0.

3. int fibonacci(int n)

If n==0 or n==1

Return 1

Else

Return Fibonacci(n-1)+Fibonacci(n-2).