|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 04**

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Data Structures Lab (CSL301) |
| Subject Professor In-charge | Prof. Swapnil S. Sonawane |
| Assisting Teachers | Prof. Swapnil S. Sonawane |

|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**Title:**

Applications of Stack ADT (Checking correctness of parenthesis)

**Objective:**

Students will be able to demonstrate the ability to analyze, design, apply and use data structures to solve engineering problems and evaluate their solutions.

**Explanation:**

1) Input the expression and put it in a character stack.

2) Scan the characters from the expression one by one.

3) If the scanned character is a starting bracket ( ‘ ( ‘ or ‘ { ‘ or ‘ [ ‘), then push it to the stack.

4) If the scanned character is a closing bracket ( ‘ ) ’ or ‘ } ’ or ‘ ] ’ ), then pop from the stack and if the popped character is the equivalent starting bracket, then proceed. Else, the expression is unbalanced.

5) After scanning all the characters from the expression, if there is any parenthesis found in the stack or if the stack is not empty, then the expression is unbalanced.

**Program Code:**

#include<stdio.h>

#include<string.h>

#define N 20

typedef struct stack

{

int a[N];

int top;

}stack;

void push(stack \*s,int x)

{

s->top++;

s->a[s->top]=x;

}

int pop(stack \*s)

{

int x;

x=s->a[s->top];

s->top--;

return x;

}

int isopenbracket(int x)

{

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

{

return 1;

}

else

{

return 0;

}

}

int isclosebracket(int x)

{

if(x=='}'||x==']'||x==')')

{

return 1;

}

else

{

return 0;

}

}

int isempty(stack \*s)

{

if(s->top==-1)

return 1;

else

return 0;

}

int checkkaro(char s1[])

{

int i,x,ele;

stack s;

s.top=-1;

for(i=0;i<strlen(s1);i++)

{

x=s1[i];

if(isopenbracket(x))

push(&s,x);

if(isclosebracket(x))

{

if(isempty(&s))

return 0;

else

{

ele=pop(&s);

if(x=='}'&&ele!='{')

return 0;

if(x==']'&&ele!='[')

return 0;

if(x==')'&&ele!='(')

return 0;

}

}

}

if(isempty(&s))

{

return 1;

}

else

{

return 0;

}

}

int main()

{

char s1[20];

printf("Enter Expression=");

gets(s1);

if(checkkaro(s1))

{

printf("\nExpression is valid...");

}

else

{

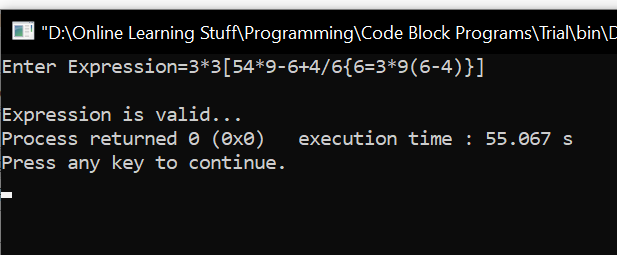
printf("\nExpression is invalid...");

}

return 0;

}

**Output:**

****

**Conclusion:**

Through this experiment, students were able to successfully apply the concept learned in order to create a program to check the correctness of parenthesis.