11/12/24

Pseudocode

1) Write a program to stimulate the working of stack using an array with the following:

(a) Push
(b) Pop
(c) Display

-> i) #include <stdio.h>
#include <math.h>
top int top = -1;
# define SIZE 5
int stack[SIZE]

ii) ~~POP~~ PUSH:

(a)
int x;
printf("Enter value: ");
scanf("%d", x);

(b)
if(top == -1){
~~printf(" ")~~
if(top >= (SIZE - 1) {
print ("overflow condition");
}
else {
top++;
stack[top] = x;
}

iii) (c) POP:

if(top == -1){ int temp;
print ("underflow condition");
}

```
else {
    temp = stack[top];
    top--;
    printf("The popped item is %d", temp);
}
```

iv) (d) display:

```
for(int i= top; i>=0 ; i=(i+1){
    printf("%d", stack[top]");
}
```

2) WAP to convert a given valid parenthesis infix arithmetic ~~infix~~ expression to postfix expression. The expression consists of single character operands and the binary operator + (plus), - (minus), * (multiply), / (divide).

-> i) #include <stdio.h>
#include <math.h>
#include <string.h>
#define MAX 100
char stack[MAX]; char infix[MAX]; char postfix[MAX];
int top = -1;

ii) PUSH:

```
int x;
if(push printf("Enter value"); scanf("%d",&x);
if (top>=(SIZE-1)){
    printf("Overflow");
}
else {
    top++;
    stack[top] = x;
}
```

iii) POP:

```
int temp;
if (top == -1) {
    printf("Underflow condition");
}
else {
    temp = stack[top]
    top--
    printf("%d", stack[top]);
    return stack[top] };
```

iv) display:

```
for (int i = top ; i>=0 ; i=i-1){
    printf("%d", stack[top]);
}
```

v) ~~infix~~ isOperator :

```
int isOperator (char a){
return ( a == '/' || a == '*' || a == '+' ||
    a == '-' );
}
```

vi) ~~prece~~ precedence :

```
int precedence (char s){
    if (s == '*' || s == '/'){
        return 2; }
    else if (s == '+' || s == '-'){
        return 1; }
    else {
        return 0;
    }
```

vii) ~~void~~ infixToPostfix (char infix[]){

```
int i=0, int j=0;

while(infix[i] != '\0'){

    if(! isOperator (infix[i]){
        postfix[j] = infix[i]
        i++;
        j++;
    }
```

© else {

while (top >=0 && precedence (stack[top]) >=
precedence (infix[i])){

postfix[j] = pop();
j++ ;
}
push (infix[i]);
i++
}
}

(d) while (top >=0) {
postfix[j] = pop();
j++ ;
}
postfix[j] = '\0';

printf(" Infix expression = %s ", infix);

printf(" Postfix expression = %s", postfix);
}

① Main function for 1st code:

```
void main(){
    int ch = -1
    while (ch != 0) {
    printf("Enter: \n1: push \n2: pop \n3: display
        \n 0: exit program");
    scanf("%d", &ch);

    Switch (ch){
      Case 1: push();
              break;
      Case 2: pop();
              break;
      Case 3: display();
              break;
      Case 0: printf("The program is terminating");
              break;
      default: printf("invalid input");
    }
  }
}
```

Output

② The infix expression is: A+B*C-D
The postfix expression is:

1> Enter:

1: push

2: pop

3: display

0#: ~~terminat~~ exit program

display

1

Enter value: 2

1

Enter value: 4

3

4 2

0

Terminating program.

inating");