28/12/23

Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands & the binary operators +(plus), -(minus), *(multiply) and ^(power).

```
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
# define MAX 100
char stack[MAX];
char infix[MAX];
char postfix[MAX];
int top=-1;
void push(char);
char pop();
int isEmpty();
void inToPost( );
void print();
int precedence(char);
int main()
{
 printf("enter infix expression:");
 gets(infix);
 inToPost();
 print();
 return 0;
}
void inToPost()
{
   int i,j=0;
   char symbol,next;
   for (i=0; i<strlen(infix); i++)
   {
     symbol = infix[i];
     switch(symbol)
     {
      case '(':
              push(symbol);
              break;
      case ')':
             while((next=pop())!='(')
            postfix[j++]=next;
            break;
    case '+':
    case '-':
    case '*':
    case '/':
    case '^':
        while (!isEmpty() && precedence(stack[top])>=
                                precedence(symbol))
                        postfix[j++]= pop();
            push(symbol);
            break;
     default:
            postfix[j++]= symbol;
     }
  }
  while (!isEmpty())
        postfix[j++]= pop();
  postfix[j]= '\0';
}
int precedence (char symbol) ——> ②
{
  switch (symbol)
  {
  case '^' :
        return 3;
  case '/' :
  case '*' :
        return 2;
  case '+' :
  case '-' :
        return 1;
  default:
        return 0;
  }
}
void print () ——> ③
{
  int i =0;
```

```
 printf ("The equivalent postfix expression is : ");
  while (postfix[i])
   {
      printf (" %c" , postfix[i++]);
    }
   printf (" \n");
}
void push (char c)
{
     if(top == Max -1)
      {
         printf (" stack overflow");
         return ;
       }
     top ++;
     stack[top]= c;
 }
 char pop ()
 {
     char c ;
     if(top == -1)
     {
        printf ("stack underflow");
        exit (1);
      }
     c = stack[top] ;
     top = top - 1;
     return c ;
 }
int    isEmpty ()
{
   if(top == -1)
        return 1;
   else return 0;
 }
```

);

output :

```
enter infix expression: a*b+c*d-e
The equivalent postfix expression is: ab*cd*+e-
```

2) postfix evaluation:

```
#include<stdio.h>
int stack[20];
int top = -1;
void push(int x)
{
 stack[++top]=x;
}
int pop()
{
 return stack[top--];
}
int main()
{
 char exp[20];
 char *e;
 int n1,n2,n3,num;
 printf("Enter the expression: ");
 scanf(" %s", exp);
 e=exp;
 while(*e != '\0')
 {
   if(isdigit(*e))
   {
    num=*e-48;
    push(num);
   }
   else
   {
    n1=pop();
    n2=pop();
    switch(*e)
    {
    case '+':
     {
      n3=n1+n2;
      break;
     }
     case '-':
     {
      n3=n2-n1;
      break;
     }
     case '*':
     {
      n3=n1*n2;
      break;
     }
```

[illegible]
18/1/24

```
        Case '/':
        {
          n3 = n2/n1;
          break;
        }
        }
        push(n3);
      }
      e++;
    }
    printf("\n The result of expression %s = %d\n\n", exp, pop());
    return 0;
  }
```

output: Enter the expression: 23*5+

```
          The result of expression 23*5+ = 11
```

3) linear Queue :-

```
#include<stdio.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = -1;
int front = -1;
main()
{
  int choice;
  while (1)
  {
   printf("1. Insert element to queue \n");
   printf("2. delete element from queue \n");
   printf("3. display all elements of queue\n");
   printf("4. Quit \n");
   printf("Enter your choice: ");
   scanf("%d", &choice : ");
   switch (choice)
   {
    case 1:
    insert();
    break;
    Case 2:
    delete();
    break;
    Case 3:
    display();
    break
    Case 4:
    exit(1);
    default
```

```
        print("wrong choice \n");
      }
    }
  }
void insert()
{
 int add_item;
 if (rear == MAX - 1)          | if((rear + 1) % MAX == front)
 printf("Queue overflow \n");
 else
 {
   if( front == -1)
   front = 0;
   printf("Insert the element in queue : ");
   scanf("%d", &add_item);
   rear = rear + 1;          % MAX ;
   queue-array[rear] = add_item;
 }
}
void delete()
{
  if( front == -1 || front > rear)
  {
   printf("Queue underflow \n");
   return;
  }
  else
  {
   printf("Element deleted from queue is: %d\n", queue-array[front]);
   front = front + 1;        | front = (front + 1) % MAX;
  }
}

void display()
{
  int i;
  if(front == -1)
     printf("Queue is empty \n");
  else
  {
   printf("Queue is :\n");
   for (i= front;  i<= rear;  i++)   for(i= front; i!=rear; i=(i+1)%MAX)
       printf(" %d", queue-array[i]);
   printf("\n");
  }
}
```

output :

```
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit

> Enter your choice : 1
Insert the element in queue : 67
> Enter your choice : 3
~~3. Display all~~
  Queue is :
  67

> Enter your choice : 2
Element deleted from queue is : 67
> Enter your choice : 4
```

[illegible]
18/1/24

18/1/24

Single linked list
[To delete at begining, end and at specific position]

```
void begin_delete()
{
 struct node *ptr;
 if(head == NULL)
  {
    printf("\nList is empty \n");
  }
 else
  {
    ptr = head;
    head = ptr -> next;
    free(ptr);
    printf("\n Node deleted from the begining \n");
  }
}
void last_delete()
{
 struct node *ptr, *ptr1;
 if(head == NULL)
  {
    printf("\n List is empty");
  }
 else if(head -> next == NULL)
 {
   head = NULL;
   free(head);
   printf("\n Only node of the list deleted \n");
 }
else
 {
   ptr = head;
   while(ptr -> next != NULL)
   {
     ptr1 = ptr;
     ptr = ptr -> next;
   }
   ptr1 -> next = NULL;
   free(ptr);
   printf("\n Deleted node from the last \n");
 }
}
```

[illegible]
18/1/24

```
void random delete ()
{
  struct node *ptr, *ptr1;
  int loc, i;
  printf(" \n Enter the location to perform deletion \n");
  scanf("%d", &loc);
   ptr = head;
   for(i=0; i<loc; i++)
   {
      ptr1 = ptr ;
      ptr = ptr->next ;
      if(ptr == NULL)
      {
        printf("\n can't delete ");
        return;
       }
   }
   ptr1 ->next = ptr->next;
   free(ptr);
   printf("\nDeleted node %d ", loc+1);
}
```

O/P :

```
elements are :
5
4
3
2
1
11
12
13
14
1. to insert at the beginning
2. to insert at the end
3. to insert at the position
4. to display
5. delete from beginning
6. delete from end
7. random delete
8. exit
enter your choice
5
-----
4
3
2
1
11
12
13
14
enter your choice
6
-----
4
3
2
1
11
12
13
-----
enter your choice
7
Enter the location of the node after which you want to perform deletion
1
-----
4
2
1
11
12
13
```