SkillRack

GAVASKAR K-2112009@nec 0/10 5 0 0 198 35801

Valid Till: 31-May-2025 Home Reports Profile Help Logout

Go Back

Completed Programs

Evaluate Postfix (Id-13772)

The program must accept a string S containing a postfix expression as the input. The program must evaluate the postfix expression and print the result as the output.
Note: The operands and operators are separated by a space.
The operators in the expression are +, -, *, / and %.

Boundary Conditions(s):
1 <= Length of S <= 100

Input Format:
The first line contains S.

Output Format:
The first line contains the result.

Example Input/Output 1:
Input:
10 20 + 2 *

Output:
60

Example Input/Output 2:
Input:
10 2 * 80 + 10 * 200 /

Output:
5

Show My Solution

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#define SIZE 50
int s[SIZE];
int top=-1;
int flag=0;
int pop()
{
    return(s[top--]);
}
int push(int elem)
{
    if(flag==1)
    {
        int num;
        num=pop();
        s[++top]=elem+10*num;
    }
    else if(flag==0)
    {
        s[++top]=elem;
        flag=1;
    }
}

int main()
{
    char pofx[50],ch;
    int i=0,op1,op2;
    scanf("%[^\n]s",pofx);
    while((ch=pofx[i++])!='\0')
    {
        if(isdigit(ch))push(ch-'0');
        else if(ch==' ')
        flag =0;
        else
        {
            flag=0;
            op2=pop();
            op1=pop();
            switch(ch)
            {
                case '+':
                push(op1+op2);
                break;
                case '-':
                push(op1-op2);
                break;
                case '*':
                push(op1*op2);
                break;
                case '/':
```

```
                push(op1/op2);
                break;
                case '%':
                push(op1%op2);
                break;
            }
        }
    }
    printf("%d",s[top]);
}
```

FCFS Scheduling (Id-13773)

N queries are passed as the input to the program. Each query is to be processed by the processor every second. Implement the FCFS (First Come First Served) schedule to schedule the processes based on the query. Each query has a character CH conditionally followed by an integer representing the process id. The query types are given below.
If the character CH is + then it is a new process which must be added to queue. The process ID will be followed by the character CH.
If the character is - then the process in the front is removed from the queue.
Assume that the process in the front is being run and the remaining processes in the queue are waiting.

The program must print the process ID, waiting time and the burst time of the processes in the FCFS order as shown in the Example Input/Output section.

Boundary Condition(s):
1 <= N, process ID <= 1000

Input Format:
The first line contains N.
The next N lines contain N queries.

Output Format:
The lines contain the process ID, waiting time and burst time of each process as shown in the Example Input/Output section.

Example Input/Output 1:
Input:
8
+ 1
+ 2
+ 3
-
-
+ 4
-
-

Ouptut:
```
1 0s 3s
2 2s 1s
3 2s 2s
4 1s 1s
```

Explanation:

WT-Waiting Time, BT-Burst Time
After 1 second, the queue status is
Process 1: WT=0 BT=1

After 2 seconds, the queue status is
Process 1: WT=0 BT=2
Process 2: WT=1 BT=0

After 3 seconds, the queue status is
Process 1: WT=0 BT=3
Process 2: WT=2 BT=0
Process 3: WT=1 BT=0

Now in the beginning of the 4th second, process 1 is removed.
Process 1: WT=0 BT=3 (Removed from queue)
After 4 seconds, the queue status is
Process 2: WT=2 BT=1
Process 3: WT=2 BT=0

Now in the beginning of the 5th second, process 2 is removed.
Process 1: WT=0 BT=3 (Removed from queue)
Process 2: WT=2 BT=1 (Removed from queue)
After 5 seconds, the queue status is
Process 3: WT=2 BT=1

Now in the beginning of the 6th second, process 4 is added.
Process 1: WT=0 BT=3 (Removed from queue)
Process 2: WT=2 BT=1 (Removed from queue)
After 6 seconds, the queue status is
Process 3: WT=2 BT=2
Process 4: WT=1 BT=0

Now in the beginning of the 7th second, process 3 is removed.
Process 1: WT=0 BT=3 (Removed from queue)
Process 2: WT=2 BT=1 (Removed from queue)
Process 3: WT=2 BT=2 (Removed from queue)
After 7 seconds, the queue status is
Process 4: WT=1 BT=1

In the beginning of the 8th second, process 4 is removed.
Process 1: WT=0 BT=3 (Removed from queue)
Process 2: WT=2 BT=1 (Removed from queue)

Process 3: WT=2 BT=2 (Removed from queue)
Process 4: WT=1 BT=1 (Removed from queue)

Hence the output is
1 0s 3s
2 2s 1s
3 2s 2s
4 1s 1s

Example Input/Output 2:
Input:
10
+ 1
+ 2
+ 3
+ 4
+ 5
-
-
-
-
-

Ouptut:
1 0s 5s
2 4s 1s
3 4s 1s
4 4s 1s
5 4s 1s

Show My Solution

```
#include<stdio.h>
#include<stdlib.h>


    int q[10001],w[1001],b[1001];
    int f=-1;
    int r=-1;
    int c=0;
    void insert(int p)
    {
        if(f==-1)
        {
            f+=1;
            r+=1;
            q[f]=p;
            w[f]=0;
            b[f]=1;
            for(int i=f;i<r;i++)
            {
                w[i]+=1;
            }
        }
        else
        {
            r+=1;
            q[r]=p;
            b[r]=0;
            b[f]+=1;
            for(int i=f+1;i<=r;i++)
            {
                if(r!=f)
                {
                    w[i]+=1;
                }
            }
        }
    }
    void pop()
    {
        int s=q[f];
        int m=w[f];
        int d=b[f];
        f+=1;
        b[f]=b[f]+1;
        for(int i=f+1;i<=r;i++)
        {
            if(r!=f)
            {
                w[i]+=1;
            }
        }
        printf("%d %ds %ds\n",s,m,d);
    }
    int main()
    {
        int n;
        char q;
        scanf("%d",&n);
        for(int i=0;i<=n;i++)
        {
            scanf("%c ",&q);
            if(q=='+')
            {
                int p;
                scanf(" %d\n",&p);
                insert(p);
                c++;
            }
            else if(q=='-')
            {
                pop();
            }
        }
    }
```

Polynomial Addition (Id-13774)

The program must accept two polynomial expressions as the input. The program must add the polynomials and print the sum as the output. The polynomials are sorted by their exponent values in descending order. The sum also must be sorted by their exponent values in descending order. Please fill in the missing lines of code so that the program runs successfully.

Boundary Condition(s):
5 <= Length of each expression <= 1000

Input Format:
The first line contains the first polynomial expression.
The second line contains the second polynomial expression.

Output Format:
The first line contains "Polynomial Addition: " followed by the sum as per the given condition.

Example Input/Output 1:
Input:
+2x^4+3x^2-1x^1
-2x^5+3x^4+2x^2-2x^1

Ouptut:
Polynomial Addition: -2x^5+5x^4+5x^2-3x^1

Example Input/Output 2:
Input:
-10x^10-2x^7+2x^4-9x^2
10x^10+5x^8-7x^7+4x^5

Ouptut:
Polynomial Addition: +5x^8-9x^7+4x^5+2x^4-9x^2 Show My Solution

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int coefficient;
    int exponent;
    struct Node *next;
};
struct Node *head=NULL,*tail=NULL;
```

```
void insert(int coefficient, int exponent)
{
    struct Node * Newnode;
    Newnode=(struct Node*)malloc(sizeof(struct Node));
    Newnode->coefficient=coefficient;
    Newnode->exponent=exponent;
    if(head==NULL)
    {
        head=tail=Newnode;
        tail->next=NULL;
    }
    else
    {
        tail->next=Newnode;
        tail=Newnode;
        tail->next=NULL;
    }
}
struct Node* getPolynomialExpression(char expression[])
{
    int lengthOfExpression=strlen(expression),flag=0,ctr=0,j=0;
    char arr[1001][1001];
    for(int i=0;i<=lengthOfExpression;i++)
    {
        if(flag==1&&!isdigit(expression[i])||expression[i]=='\0')
        {
            arr[ctr][j]='\0';
            ctr++;
            j=0;
            if(expression[i]=='+'||expression[i]=='-')
            {
                arr[ctr][j]=expression[i];
                j++;
            }
            flag=0;
        }
        else if((expression[i]=='+'||expression[i]=='-')||isdigit(expression[i]))
        {
            arr[ctr][j]=expression[i];
            j++;
            flag=1;
        }
    }
    for(int i=0;i<ctr;i+=2)
    {
        insert(atoi(arr[i]),atoi(arr[i+1]));
    }
    struct Node *tmp=head;
    head=NULL;
    return tmp;
}
struct Node* polynomialAddition(struct Node *poly1,struct Node *poly2)
{
    struct Node *ptr1=poly1, *ptr2=poly2;
```

```
        while(ptr1&&ptr2)
        {
            if(ptr1->exponent==ptr2->exponent)
            {
                int coeff=ptr1->coefficient+ptr2->coefficient;
                if(coeff!=0)
                {
                    insert(coeff,ptr1->exponent);
                }
                ptr1=ptr1->next;
                ptr2=ptr2->next;
            }
            else if(ptr1->exponent>ptr2->exponent)
            {
                insert(ptr1->coefficient,ptr1->exponent);
                ptr1=ptr1->next;
            }
            else
            {
                insert(ptr2->coefficient,ptr2->exponent);
                ptr2=ptr2->next;
            }

        }
        while(ptr1!=NULL||ptr2!=NULL)
        {
            if(ptr1)
            {
                insert(ptr1->coefficient,ptr1->exponent);
                ptr1=ptr1->next;
            }
            if(ptr2)
            {
                insert(ptr2->coefficient,ptr2->exponent);
                ptr2=ptr2->next;
            }
        }
    return head;
    }
    void printPolynomial(struct Node *exp)
    {
        while(exp)
        {
            printf("%+dx^%d", exp->coefficient, exp->exponent);
            exp = exp->next;
        }
    }

    int main()
    {
        char polynomialStr1[1001], polynomialStr2[1001];
        scanf("%s %s", polynomialStr1, polynomialStr2);

        struct Node *exp1 = getPolynomialExpression(polynomialStr1); //head of exp1
        struct Node *exp2 = getPolynomialExpression(polynomialStr2); //head of exp2
        struct Node *result = polynomialAddition(exp1, exp2);

        printf("Polynomial Addition: ");
        printPolynomial(result);
    }
```

Positive Front & Negative Rear - Queue (Id-13905)

The program must accept **N integers** as the input. For each integer **X** among the N integers, the program must perform the following operations and form a queue of integers.
- If X is a **positive** integer, the program must **add X** at the **beginning** of the queue.
- If X is a **negative** integer, the program must **add X** at the **end** of the queue.
- If X is **0**, the program must **remove** exactly one integer from the **beginning** and the **end** of queue (if present).
After performing each operation the program must print all the integers in the queue from the beginning to the end. If queue is empty, then the program must print **-1** as the output.

Boundary Condition(s):
1 <= N <= 100

Input Format:
The first line contains N.
The second line contains N integers separated by a space.

Output Format:
The first line contains the integer value(s) separated by a space.

Example Input/Output 1:
Input:
10
88 63 -13 55 -98 -10 -11 0 0 50

Output:
```
88
63 88
63 88 -13
55 63 88 -13
55 63 88 -13 -98
55 63 88 -13 -98 -10
55 63 88 -13 -98 -10 -11
63 88 -13 -98 -10
88 -13 -98
50 88 -13 -98
```

Explanation:
88 is a **positive** integer. Queue: **88**
63 is a **positive** integer. Queue: **63 88**
-13 is a **negative** integer. Queue: **63 88 -13**

55 is a **positive** integer. Queue: **55 63 88 -13**
-98 is a **negative** integer. Queue: **55 63 88 -13 -98**
-10 is a **negative** integer. Queue: **55 63 88 -13 -98 -10**
-11 is a **negative** integer. Queue: **55 63 88 -13 -98 -10 -11**
0 is neither positive nor negative. Queue: **63 88 -13 -98 -10**
0 is neither positive nor negative. Queue: **88 -13 -98**
50 is a **positive** integer. Queue: **50 88 -13 -98**

Example Input/Output 2:
Input:
8
20 -15 -20 0 0 -10 -30 40

Output:
20
20 -15
20 -15 -20
-15
-1
-10
-10 -30
40 -10 -30

Example Input/Output 3:
Input:
5
0 13 14 0 0

Output:
-1
13
14 13
-1
-1

Show My Solution

```
#include<stdio.h>
#include<stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};
struct Node *head=NULL;
struct Node *tail=NULL;
int isempty()
{
    if(head==NULL)
    return 1;
```

```
        else
        return 0;
    }
    void insert_begin(int d)
    {
        struct Node *newnode;
        newnode=(struct Node*)malloc(sizeof(struct Node));
        newnode->data=d;
        if(head==NULL)
        {
            head=newnode;
            tail=newnode;
            newnode->next=NULL;
        }
        else
        {
            newnode->next=head;
            head=newnode;
        }
    }
    void insert_end(int d)
    {
        struct Node *temp;
        temp=(struct Node*)malloc(sizeof(struct Node));
        temp->data=d;
        if(head==NULL)
        {
            tail=temp;
            head=temp;
            tail->next=NULL;
        }
        else
        {
            tail->next=temp;
            tail=temp;
            tail->next=NULL;
        }
    }
    int delete_be()
    {
        if((head==tail&&head==NULL)||(head==tail&&head!=NULL)){
            if(head==tail&&head!=NULL)
            {
                struct Node *temp;
                temp=head;
                head=tail=NULL;
                free(temp);
            }
        }else
        {
            if(head!=NULL)
            {
                struct Node *temp;
                temp=head;
                while(temp->next!=tail)
                {
                    temp=temp->next;
                }
                tail=temp;
                tail->next=NULL;
                free(temp->next);
                struct Node *ptr;
                ptr=head;
                head=ptr->next;
                free(ptr);
            }
        }
    }
    void display()
    {
        struct Node *temp;
        temp=head;
        if(head==NULL){
            printf("-1\n");
        }else{
            while(temp!=NULL)
            {
                printf("%d ",temp->data);
                temp=temp->next;
            }
            printf("\n");
        }
    }
    int main()
    {
        int N,data;
        scanf("%d",&N);
        for(int i=0;i<N;i++)
        {
            scanf("%d",&data);
            if(data>0)
            {
                insert_begin(data);
            }else if(data<0)
            {
                insert_end(data);
            }else{
                delete_be(data);
            }
            display();
        }
    }
```