

# PS1

● Graded

## Student

SRUTHI SUBRAMANIAN

## Total Points

100 / 100 pts

## Autograder Score

100.0 / 100.0

## Passed Tests

Test 1 (10/10)

Test 2 (10/10)

Test 3 (10/10)

Test 4 (10/10)

Test 5 (20/20)

Test 6 (20/20)

Test 7 (20/20)

## Autograder Results

Test 1 (10/10)

Test 2 (10/10)

Test 3 (10/10)

Test 4 (10/10)

Test 5 (20/20)

Test 6 (20/20)

Test 7 (20/20)

## Submitted Files

```
1  #include <stdio.h>
2
3  int main() {
4      int prob;
5      scanf("%d",&prob);
6      if (prob==0){
7          //calculator. priority is -of the number assigned.
8          int n;
9          scanf("%d",&n);
10         float power(float a, float b){
11             //raise a to b
12             float final=1;
13             for (int i=1;i<=b;i++){
14                 final=final*a;
15             }
16             return final;
17         }
18         float op(float num1, float num2, float op){
19             if (op==1){
20                 return (num1-num2);
21             }
22             else if (op==2){
23                 return (num1+num2);
24             }
25             else if (op==3){
26                 return (num1*num2);
27             }
28             else if (op==4){
29                 return (num1/num2);
30             }
31             else if (op==5){
32                 return (power(num1,num2));
33             }
34             else {
35                 return 0;
36             }
37         }
38         float Calculate(float* array, int length){
39             //takes in the array of integers, length is the length of it. now calculate the value of this string
40             (this has no paranthesis!!)
41             float numbers[length];
42             int operators[length];
43             int num=0;
44             int o=0;
45             int i,n1,n2;
46             float val;
47             for (i=0;i<length;i++){
48                 if (array[i]<0){
49                     operators[o]=array[i];
```

```

49         o=o+1;
50     }
51     else {
52         numbers[num]=array[i];
53         num=num+1;
54     }
55 }
56 //num is the number of numbers and o is the number of operators in the array
57 for (i=o-1;i>-1;i--){
58     if (operators[i]==-5){
59         val=op(numbers[i], numbers[i+1], -5);
60         numbers[i]=val;
61         numbers[i+1]=0;
62         operators[i]=0;
63     }
64 }
65 //clear zeroes, update numbers, operators, o and num, then do /. Similarly for *, + and - (do +
and - together)
66 int newnums[num];
67 int newops[o];
68 n1=0;
69 n2=0;
70 for (i=0;i<num;i++){
71     if (numbers[i]!=0){
72         newnums[n1]=numbers[i];
73         n1=n1+1;
74     }
75 }
76 for (i=0;i<o;i++){
77     if (operators[i]!=0){
78         newops[n2]=operators[i];
79         n2=n2+1;
80     }
81 }
82 num=n1;
83 o=n2;
84 for (i=0;i<n1;i++){
85     numbers[i]=newnums[i];
86 }
87 for (i=0;i<n2;i++){
88     operators[i]=newops[i];
89 }
90 //DIVISION
91 for (i=0;i<o;i++){
92     if (operators[i]==-4){
93         val=op(numbers[i], numbers[i+1], -4);
94         numbers[i+1]=val;
95         numbers[i]=0;
96         operators[i]=0;
97     }
98 }
99 n1=0;

```

```
100     n2=0;
101     for (i=0;i<num;i++){
102         if (numbers[i]!=0){
103             newnums[n1]=numbers[i];
104             n1=n1+1;
105         }
106     }
107     for (i=0;i<o;i++){
108         if (operators[i]!=0){
109             newops[n2]=operators[i];
110             n2=n2+1;
111         }
112     }
113     num=n1;
114     o=n2;
115     for (i=0;i<n1;i++){
116         numbers[i]=newnums[i];
117     }
118     for (i=0;i<n2;i++){
119         operators[i]=newops[i];
120     }
121     //MULTIPLICATION
122     for (i=0;i<o;i++){
123         if (operators[i]==-3){
124             val=op(numbers[i], numbers[i+1], -3);
125             numbers[i+1]=val;
126             numbers[i]=0;
127             operators[i]=0;
128         }
129     }
130     n1=0;
131     n2=0;
132     for (i=0;i<num;i++){
133         if (numbers[i]!=0){
134             newnums[n1]=numbers[i];
135             n1=n1+1;
136         }
137     }
138     for (i=0;i<o;i++){
139         if (operators[i]!=0){
140             newops[n2]=operators[i];
141             n2=n2+1;
142         }
143     }
144     num=n1;
145     o=n2;
146     for (i=0;i<n1;i++){
147         numbers[i]=newnums[i];
148     }
149     for (i=0;i<n2;i++){
150         operators[i]=newops[i];
151     }
```

```

152 //ADDITION AND SUBTRACTION
153 for (i=0;i<o;i++){
154     val=op(numbers[i], numbers[i+1], operators[i]);
155     numbers[i+1]=val;
156     numbers[i]=0;
157     operators[i]=0;
158 }
159 n1=0;
160 n2=0;
161 for (i=0;i<num;i++){
162     if (numbers[i]!=0){
163         newnums[n1]=numbers[i];
164         n1=n1+1;
165     }
166 }
167 for (i=0;i<o;i++){
168     if (operators[i]!=0){
169         newops[n2]=operators[i];
170         n2=n2+1;
171     }
172 }
173 num=n1;
174 o=n2;
175 for (i=0;i<n1;i++){
176     numbers[i]=newnums[i];
177 }
178 for (i=0;i<n2;i++){
179     operators[i]=newops[i];
180 }
181 return numbers[0];
182
183 }
184
185 float array[n];
186 int elem;
187 int num=n;
188 int parastart[num];
189 int paraend[num];
190 int ordered[num];
191 int i,j, num1, op1, p,q;
192 for (i=0;i<n;i++){
193     scanf("%d",&elem);
194     array[i]=elem;
195 }
196 num1=0;
197 op1=0;
198 p=0; //number of paranthesis
199 q=0;
200 for (i=0;i<num;i++){
201     if (array[i]<0){
202         if (array[i]==-6){
203             parastart[p]=i;

```

```

204         p=p+1;
205     }
206 }
207 }
208 for (i=0;i<num;i++){
209     if (array[i]<0){
210         if (array[i]==-7){
211             ordered[p-q-1]=i;
212             q=q+1;
213         }
214     }
215 }
216 for (i=p-1;i>-1;i--){
217     elem=parastart[i];
218     for(j=q-1;j>-1;j--){
219         if (ordered[j]!=0){
220             if (ordered[j]>elem){
221                 paraend[i]=ordered[j];
222                 ordered[j]=0;
223                 break;
224             }
225         }
226     }
227 }
228
229 //simplify the paranthesis p number of times
230 int s,e;
231 float val;
232 float newarr[num];
233 int len;
234 while (p>0){
235     len=0;
236     s=parastart[p-1];
237     e=paraend[p-1];
238     p=p-1;
239     for (i=s+1;i<e+1;i++){
240         if (array[i]!=0){
241             newarr[len]=array[i];
242             len=len+1;
243         }
244     }
245     len=len-1;
246     val=Calculate(newarr,len);
247     int v=val;
248     array[s]=val;
249     for (i=s+1;i<e+1;i++){
250         array[i]=0;
251     }
252 }
253 //remove all zeroes.
254 len=0;
255 for (i=0;i<n;i++){

```

```

256     if (array[i]!=0){
257         newarr[len]=array[i];
258         len=len+1;
259     }
260 }
261
262 val=Calculate(newarr, len);
263 int answer;
264 answer =val;
265 printf("%d", answer);
266
267 }
268 else if (prob==1){
269     //matrix multiplication
270     int n;
271     scanf("%d",&n);
272     int dim[n];
273     int a,min;
274     for (int i=0;i<n;i++){
275         scanf("%d",&a);
276         dim[i]=a;
277     }
278     int MAT[n][n];
279     int i,j,k,l,j;
280     for (i=0;i<n-1;i++){
281         for (j=0;j<n-i-1;j++){
282             if (i==0){
283                 MAT[j][j]=0;
284             }
285             else if (i==1){
286                 MAT[j][j+i]=dim[j]*dim[j+1]*dim[j+2];
287             }
288             else{
289                 l=j;
290                 j=i+j;
291                 min=MAT[l][l]+MAT[l+1][j]+dim[l]*dim[l+1]*dim[j+1];
292                 for (k=l+1;k<j;k++){
293                     a=MAT[l][k]+MAT[k+1][j]+dim[l]*dim[k+1]*dim[j+1];
294                     if (a<min){
295                         min=a;
296                     }
297                 }
298                 MAT[l][j]=min;
299             }
300         }
301     }
302     printf("%d",MAT[0][n-2]);
303 }
304
305 return 0;
306 }

```