

PS1

● Graded

Student

ANSHUMAN SINGH

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  #include <stdlib.h>
3  #include <limits.h>
4
5
6  typedef struct{
7      int *arr;
8      int top;
9      int max;
10 } Stack;
11
12 void initStack(Stack *stack, int max){
13     stack->arr = (int*)malloc(max*sizeof(int));
14     stack->top = -1;
15     stack->max = max;
16 }
17
18 void freeStack(Stack *stack){
19     free(stack->arr);
20 }
21
22 void push(Stack *stack, int val){
23     if (stack->top>=stack->max-1)
24         exit(0);
25     stack->arr[++stack->top]=val;
26 }
27
28 int pop(Stack *stack){
29     if (stack->top>=0)
30         return stack->arr[stack->top--];
31     return 0;
32 }
33
34 int peek(Stack *stack){
35     if (stack->top>=0)
36         return stack->arr[stack->top];
37     return 0;
38 }
39
40 int inPriority(int op){
41     switch (op){
42         case -1: return 1;
43         break;
44         case -2: return 1;
45         break;
46         case -3: return 2;
47         break;
48         case -4: return 4;
49         break;
```

```
50     case -5: return 5;
51     break;
52     case -6: return 0;
53     break;
54     default: return -1;
55 }
56 }
57
58 int outPriority(int op){
59     switch (op){
60         case -1: return 1;
61         break;
62         case -2: return 1;
63         break;
64         case -3: return 2;
65         break;
66         case -4: return 3;
67         break;
68         case -5: return 6;
69         break;
70         default: return -1;
71     }
72 }
73
74 int power(int num1, int num2){
75     int k=num1;
76     while(num2-->1)
77         k*=num1;
78     return k;
79 }
80
81 int operate(int num1, int num2, int operator){
82     int result;
83     switch (operator){
84         case -1: return num1-num2;
85         break;
86         case -2: return num1+num2;
87         break;
88         case -3: return num1*num2;
89         break;
90         case -4: return num1/num2;
91         break;
92         case -5: return power(num1,num2);
93         break;
94         default: return 0;
95     }
96 }
97
98 void popNoperate(Stack *numStack, Stack *opStack){
99     int operator = pop(opStack);
100     int num2 = pop(numStack);
101     int num1 = pop(numStack);
```

```

102     int result = operate(num1, num2, operator);
103     push(numStack, result);
104 }
105
106 void Calculator(){
107     int n;
108     scanf("%d",&n);
109     Stack opStack, numStack;
110     initStack(&opStack, n);
111     initStack(&numStack, n);
112     while(n-->0){
113         int x;
114         scanf("%d",&x);
115         if(x==6){
116             push(&opStack, x);
117         }
118         else if(x==7){
119             while(peek(&opStack)!=6){
120                 popNoperate(&numStack, &opStack);
121             }
122             pop(&opStack);
123         }
124         else if(x<0){
125             while(opStack.top!=1&&outPriority(x)<=inPriority(peek(&opStack))){
126                 popNoperate(&numStack, &opStack);
127             }
128             push(&opStack, x);
129         }
130
131         else
132             push(&numStack, x);
133     }
134
135     while(opStack.top>=0){
136         popNoperate(&numStack, &opStack);
137     }
138
139     printf("%d",pop(&numStack));
140     freeStack(&opStack);
141     freeStack(&numStack);
142 }
143
144
145 void MinMult(){
146     int n;
147     scanf("%d",&n);
148     int *arr=(int*)malloc(n*sizeof(int));
149     for(int i=0;i<n;i++)
150         scanf("%d",&arr[i]);
151     int **M= (int**)malloc(n*sizeof(int*));
152     for(int i = 0; i < n; i++){
153         M[i] = (int*)malloc(n*sizeof(int));

```

```
154     }
155     for(int i=0;i<n;i++){
156         for(int j=0;j<n;j++){
157             M[i][j] = 0;
158         }
159     }
160     for(int s=2;s<n;s++){
161         for(int i=0;i<n-s;i++){
162             int j=i+s;
163             M[i][j]=INT_MAX;
164             for(int k=i+1;k<j;k++){
165                 int cost = M[i][k] + M[k][j] + arr[i]*arr[k]*arr[j];
166                 if(cost<M[i][j]){
167                     M[i][j]=cost;
168                 }
169             }
170         }
171     }
172     printf("%d",M[0][n-1]);
173 }
174
175
176 int main() {
177     int option;
178     scanf("%d",&option);
179     if(option==0)
180         Calculator();
181     else if(option==1)
182         MinMult();
183     return 0;
184 }
185
```
