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

hw.c Download

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


typedef struct{
    int *arr;
    int top;
    int max;
} Stack;

void initStack(Stack *stack, int max){
    stack->arr = (int*)malloc(max*sizeof(int));
    stack->top = -1;
    stack->max = max;
}

void freeStack(Stack *stack){
    free(stack->arr);
}

void push(Stack *stack, int val){
    if (stack->top>=stack->max-1)
        exit(0);
    stack->arr[++stack->top]=val;
}

int pop(Stack *stack){
    if (stack->top>=0)
        return stack->arr[stack->top--];
    return 0;
}

int peek(Stack *stack){
    if (stack->top>=0)
        return stack->arr[stack->top];
    return 0;
}

int inPriority(int op){
    switch (op){
        case -1: return 1;
        break;
        case -2: return 1;
        break;
        case -3: return 2;
        break;
        case -4: return 4;
        break;
        case -5: return 5;
        break;
        case -6: return 0;
        break;
        default: return -1;
    }
}

int outPriority(int op){
    switch (op){
        case -1: return 1;
        break;
        case -2: return 1;
        break;
        case -3: return 2;
        break;
        case -4: return 3;
        break;
        case -5: return 6;
        break;
        default: return -1;
    }
}

int power(int num1, int num2){
    int k=num1;
    while(num2-->1)
    k*=num1;
    return k;
}

int operate(int num1, int num2, int operator){
    int result;
    switch (operator){
        case -1: return num1-num2;
        break;
        case -2: return num1+num2;
        break;
        case -3: return num1*num2;
        break;
        case -4: return num1/num2;
        break;
        case -5: return power(num1,num2);
        break;
        default: return 0;
    }
}

void popNoperate(Stack *numStack, Stack *opStack){
    int operator = pop(opStack);
    int num2 = pop(numStack);
    int num1 = pop(numStack);
```

```c
    int result = operate(num1, num2, operator);
    push(numStack, result);
}

void Calculator(){
    int n;
    scanf("%d",&n);
    Stack opStack, numStack;
    initStack(&opStack, n);
    initStack(&numStack, n);
    while(n-->0){
        int x;
        scanf("%d",&x);
        if(x==-6){
                push(&opStack, x);
            }
        else if(x==-7){
            while(peek(&opStack)!=-6){
                popNoperate(&numStack, &opStack);
            }
            pop(&opStack);
            }
        else if(x<0){
            while(opStack.top!=-1&&outPriority(x)<=inPriority(peek(&opStack))){
               popNoperate(&numStack, &opStack);
            }
            push(&opStack, x);
        }

        else
            push(&numStack, x);

    }
    while(opStack.top>=0){
        popNoperate(&numStack, &opStack);
    }

    printf("%d",pop(&numStack));
    freeStack(&opStack);
    freeStack(&numStack);
}


void MinMult(){
    int n;
    scanf("%d",&n);
    int *arr=(int*)malloc(n*sizeof(int));
    for(int i=0;i<n;i++)
    scanf("%d",&arr[i]);
    int **M= (int**)malloc(n*sizeof(int*));
    for(int i = 0; i < n; i++){
        M[i] = (int*)malloc(n*sizeof(int));
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            M[i][j] = 0;
        }
    }
    for(int s=2;s<n;s++){
        for(int i=0;i<n-s;i++){
            int j=i+s;
            M[i][j]=INT_MAX;
            for(int k=i+1;k<j;k++){
                int cost = M[i][k] + M[k][j] + arr[i]*arr[k]*arr[j];
                if(cost<M[i][j]){
                    M[i][j]=cost;
                }
            }
        }
    }
    printf("%d",M[0][n-1]);
}


int main() {
   int option;
   scanf("%d",&option);
   if(option==0)
   Calculator();
   else if(option==1)
   MinMult();
    return 0;
}

```