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

struct stack
{
    int top;
    int capacity;
    int *array;
};


struct advancedStack
{
    struct stack *elementstack;
    struct stack *minstack;
};

struct stack *createStack(int c)
{
    struct stack *S=malloc(sizeof( struct stack));
    S->top=-1;
    S->capacity=c;
    S->array=malloc(S->capacity*sizeof(int));

    return S;
}

struct advancedStack *createAdvancedStack()
{
    struct advancedStack *As=(struct advancedStack *)malloc(sizeof(struct
advancedStack));
    As->elementstack=createStack(5);
    As->minstack=createStack(5);
    return As;
}

int isEmpty(struct stack *S)
{
    return(S->top==-1);
}

int isFull(struct stack *S)
{
    return(S->top==S->capacity-1);
}


void push(struct stack *S,int k)
{
    if(isFull(S))
    {
        printf("OverFlow");
        return;
    }



    else



        S->array[++S->top]=k;


}

```

```c
int pop(struct stack *S)
{
    int data;
    if(isEmpty(S))
        {
            printf("UnderFlow");
        }


    else
    {
        data=S->array[S->top--];
        return data;
    }

}

void pushA(struct advancedStack *S,int data)
{
    if(isFull(S->elementstack))
    {
        printf("overflow, cant insert any more values");
    }

    else
    {
        push(S->elementstack,data);
        if(isEmpty(S->minstack)||data<=S->minstack->array[S->minstack->top])
            push(S->minstack,data);

    }


}

void popA(struct advancedStack *S)
{
    int data;
    if(isEmpty(S->elementstack))
        printf("underflow, cant delete any more values");
    else
    {
         data=pop(S->elementstack);
    }

    if(S->minstack->array[S->minstack->top]==data)
        pop(S->minstack);
}

int getMinimum(struct advancedStack *S)
{
    return S->minstack->array[S->minstack->top];
}

void main()
{
    struct advancedStack *S=createAdvancedStack();
    pushA(S,5);
    pushA(S,6);
    pushA(S,4);
    pushA(S,3);
    pushA(S,1);
    pushA(S,0);
   // popA(S);
    //popA(S);
    printf("Minimum is: %d",getMinimum(S));
}
```