Name:…… Q11

CS2123 Data Structures

Recall the stack ADT and its improved implementation using dynamic array with initial size of 100 elements. It was doubling the size of the dynamic array by calling Expand(stack) in Push(stack, element) function when the current stack is full.

Suppose we want to implement a similar (but an opposite) function **Shrink(stack)** that will be called in Pop(stack) when the number of elements in stack is less than the quarter of the current size of the array and the size of the array is larger than 200. Below are some of the original declarations in stack.h and stack.c files, and the modified Pop(stack) function.

```
/**** stack.h ****/

typedef double
    stackElementT;

typedef struct stackCDT
   *stackADT;

/* exported functions */
```

```
#define InitialStackSize100                    /**** improved stack.c ****/
struct stackCDT{
   stackElementT *elements;
   int count;
   int size;
};
stackElementT Pop(stackADT stack)
{
  if (StackIsEmpty(stack)) Error("Pop of an empty stack");
  if(stack->count < stack->size / 4 && stack->size >= 200)  Shrink(stack);
  return (stack->elements[--stack->count]);
}
```

So, you will **just implement Shrink(stackADT stack)** function, which should allocate a new array with the half size of the current array, copy the existing elements from old array into new array, free the old array, and update the necessary fields in the stackCDT pointed by stack.

```
static void Shrink(stackADT stack)
{
```