Sort stack using temporary stack

```
#include <stdio.h>
#include <stdlib.h>
struct stack
{
   int data;
   struct stack *next;
};
void initStack(struct stack **s)
{
   *s = NULL;
}
int isEmpty(struct stack *s)
{
   if (s == NULL)
      return 1;
   return 0;
}
void push(struct stack **s, int x)
{
   struct stack *p = (struct stack *)malloc(sizeof(*p));

   if (p == NULL)
   {
      fprintf(stderr, "Memory allocation failed.\n");
```

```
        return;
    }

    p->data = x;
    p->next = *s;
    *s = p;
}
int pop(struct stack **s)
{
    int x;
    struct stack *temp;

    x = (*s)->data;
    temp = *s;
    (*s) = (*s)->next;
    free(temp);

    return x;
}
int top(struct stack *s)
{
    return (s->data);
}
void sortedInsert(struct stack **s, int x)
{
```

```
    if (isEmpty(*s) || x < top(*s))
    {
        push(s, x);
        return;
    }

    int temp = pop(s);
    sortedInsert(s, x);
    push(s, temp);
}

void sortStack(struct stack **s)
{
    if (!isEmpty(*s))
    {
        int x = pop(s);

        sortStack(s);

        sortedInsert(s, x);
    }
}

void printStack(struct stack *s)
{
```

```
    while (s)
    {
        printf("%d ", s->data);
        s = s->next;
    }
    printf("\n");
}

int main(void)
{
    struct stack *top;

    initStack(&top);
    push(&top, 75);
    push(&top, 93);
    push(&top, 28);
    push(&top, 67);
    push(&top, 35);
    push(&top, 41);

    printf("Stack elements before sorting:\n");
    printStack(top);

    sortStack(&top);
    printf("\n\n");
```

```
    printf("Stack elements after sorting:\n");
    printStack(top);

    return 0;
}
```