Code Analysis of ChangePatties

TCSS 342: Data Structures
Jesse Bannon

Functions called in ChangePatties

Ingredient getIngredientString(char ingStr)*

Description: getIngredientString compares string ***ingStr*** against a max of 17 other strings (lower-cased names of ingredients) to return the appropriate Ingredient enum.

Value: $c_{ingString}$ (constant)

*void initializeStack(MyStack** stack)*

Description: initializeStack allocates memory to create a MyStack structure pointed by the ***stack*** parameter.

Value: $c_{initStack}$ (constant)

*int size(MyStack** stack)*

Description: Returns the integer size stored within **stack**.

Value: c_{size} (constant)

*void push(MyStack** stack, const Ingredient ingredient)*

Description: Increments the size variable within **stack** and creates and allocates memory to create a new node. The node's ingredient enum is assigned to the parameter **ingredient**. The new node's next pointer is then pointed to **stack**'s top, then top is pointed to the new node.

Value: c_{push} (constant)

*Ingredient pop(MyStack** stack)*

Description: Declares an Ingredient enum *thePop* that will be returned at the end of the function. If **stack** is not empty it will assign **stack**'s top node's ingredient to *thePop*, decrement **stack**'s size, and point **stack**'s top pointer to the top node's next. Otherwise it will assign *thePop* to 0. *thePop* is then returned.

Value: c_{pop} (constant)

*Ingredient peek(MyStack** stack)*

Description: Returns stack's top node's ingredient.

Value: c_{peek} (constant)

Runtime of ChangePatties

1	void changePatties(Burger** burger, char* pattyType)		2
	{		
2	Ingredient newPatty, tempIng;		2
3	MyStack* tempStack;		1
4	newPatty = ***getIngredient(pattyType)***;		$1 + c_{ingString}$
5	***initializeStack(&tempStack)***;		$c_{initStack}$
6	while(***size(&(*burger)->burgerStack)*** > 0)	LOOP	$c_{size} + 1$
	{		
7	tempIng = ***peek(&(*burger)->burgerStack)***;		$1 + c_{peek}$
8	if (tempIng == BEEF		3
	|| tempIng == CHICKEN		
	|| tempIng == VEGGIE)		
	{		
9	***pop(&(*burger)->burgerStack)***;		c_{pop}
10	***push(&tempStack, newPatty)***;		c_{push}
	} else		
11	***push(&tempStack, pop(&(*burger)->burgerStack))***;		$c_{push} + c_{pop}$
	}	END	
12	while(***size(&tempStack)*** > 0)	LOOP	$c_{size} + 1$
	{		
13	***push(&(*burger)->burgerStack, pop(&tempStack))***;		$c_{push} + c_{pop}$
	}	END	
14	free(tempStack);		1
	}		

The value of each line is denoted by $[value]_{line\ number}$
Let **n** equal the amount of ingredients within burger's stack.
Let **f(n)** represent the function changePatties

$$\mathbf{f(n)} = [2]_1 + [2]_2 + [1]_3 + [1 + c_{ingString}]_4 + [c_{intStack}]_5 + \mathbf{n(}[c_{size} + 1]_6 + [1+c_{peek}]_7 + [3]_8 + \mathbf{(}[c_{pop}]_9 + [c_{push}]_{10}\mathbf{)}\ \text{OR}\ \mathbf{(}[c_{push} + c_{pop}]_{11}\mathbf{))} + \mathbf{n(}[c_{size} + 1]_{12} + [c_{push} + c_{pop}]_{13}\mathbf{)} + [1]_{14}$$

$$= 6 + c_{ingString} + c_{intStack} + \mathbf{n(}5 + c_{size} + c_{peek} + c_{push} + c_{pop}\mathbf{)} + \mathbf{n(}1 + c_{size} + c_{push} + c_{pop}\mathbf{)} + 1$$
$$= 7 + c_{ingString} + c_{intStack} + \mathbf{n(}6 + c_{peek} + 2\mathbf{(}c_{size} + c_{push} + c_{pop}\mathbf{))}$$

Let $\mathbf{c_1}$ equal $7 + c_{ingString} + c_{intStack}$
Let $\mathbf{c_2}$ equal $6 + c_{peek} + 2\mathbf{(}c_{size} + c_{push} + c_{pop}\mathbf{)}$

$\mathbf{f(n) = c_1 + c_2 n}$ **f(n) € O(n)**