//Hocam I made two projects, but I submitted like how Meryem hoca said. So, you need to add #include stackStatic.c probably. The code works without error, tested.

```
//main
#include <stdio.h>
#include <string.h>
#include "stackStatic.h"

#define MAXSIZE 50

struct StackRecord{
        int topOfStack;
        char *array;
        int capacity;
};
typedef struct StackRecord* Stack;



int main(){
        Stack firstHalf;
        char phrase[MAXSIZE];

   printf("Please enter a phrase to find out if it is palindrome or not!\n");
        gets(phrase);
        firstHalf = CreateStack(strlen(phrase));
        IsPalindrome(phrase, firstHalf);

        return 0;
}
```

```
//stackStatic.c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include "stackStatic.h"

#define MAXSIZE 50

struct StackRecord{
	int topOfStack;
	char *array;
	int capacity;
};
typedef struct StackRecord* Stack;

void IsPalindrome(char phrase[MAXSIZE], Stack firstHalf){
	char ordered[MAXSIZE]; // I will make the all the phrase's letters bigger and put them in this char array.
	char firstOrdered[MAXSIZE/2], secondOrdered[MAXSIZE/2];
	Stack otherHalf, temp;
	int i = 0, j = 0, counter = 0;
	otherHalf = CreateStack(strlen(phrase)/2); // this will be the second half of the phrase, I will use temporary stack to reverse it and compare with s

	while(phrase[i] != '\0'){
		if (phrase[i] >= 65 && phrase[i] <= 90){
			ordered[j] = phrase[i];
```

```
                j++;
                counter++;
        }
        else if (phrase[i] >= 97 && phrase[i] <= 122) {
                ordered[j] = toupper(phrase[i]);
                j++;
                counter++;// this counter for the length of the array.
        }

        i++;
}
ordered[j] = '\0'; // now I created my array with all the only letters with all upper letters.



for(i=0; i<counter/2 ;i++) // this for loop for the first half.
        PushStack(ordered[i], firstHalf);

for(i=counter/2; i<counter ;i++) // this for loop for the second half.
        PushStack(ordered[i], otherHalf);

temp = CreateStack(strlen(phrase)/2); // this stack is temporary, I will pop from the otherHalf and push to temp so that I can have the same order.

while(!IsEmptyStack(otherHalf)) // push from the other half to temp
        PushStack(PopStack(otherHalf), temp);

if (counter % 2 == 1)
        PopStack(temp); // this pop operation for getting rid of the middle element if the phrase is a odd number.
```

```
	for(i=0; i<counter/2 ;i++) // I popped from first stack and send it to firstOrdered array.
		firstOrdered[i] = PopStack(firstHalf);

	j = 0;
	for(i=counter/2; i<counter ;i++){ // I popped from temp stack and send it to secondOrdered array.
		secondOrdered[j] = PopStack(temp);
		j++;
	}

	int flag = 1; // this is a flag for comparision two arrays
	for(i=0; i<counter/2 ;i++){
		if(firstOrdered[i] != secondOrdered[i]){
			flag = 0;
			break;
		}
	}

	if(flag == 1)
		printf("The phrase is palindrome!\n");
	else
		printf("The phrase is not palindrome!\n");
}

Stack CreateStack(int maxElements){
	Stack s;

	s = (struct StackRecord*)malloc(sizeof(struct StackRecord));
	if (s == NULL){
```

```
		printf("Out of memory!\n");
		exit(-1);
	}

	s->array=(char*)malloc(maxElements*sizeof(char));
	if (s->array == NULL){
		printf("Out of memory!\n");
		exit(-1);
	}

	s->capacity = MAXSIZE;
	s->topOfStack = -1;

	return s;
}

void PushStack(char x, Stack s){
	s->array[++s->topOfStack] = x;
}

int PopStack(Stack s){
	if(!IsEmptyStack(s))
		return s->array[s->topOfStack--];
}

int IsEmptyStack(Stack s){
	return s->topOfStack == -1;
}
```

```
//stackStatic.h
typedef struct StackRecord *Stack;

void IsPalindrome(char *, Stack s);
struct StackRecord *CreateStack(int);
void PushStack(char , Stack s);
int PopStack(Stack);
int IsEmptyStack(Stack);
```

```
//mainDynamic.c
#include <stdio.h>
#include <string.h>
#include "stackDynamic.h"

#define MAXSIZE 50

struct Node{
        char c;
        struct Node *next;
};
typedef struct Node StackRecord;
typedef StackRecord *Stack;



int main(){
        Stack firstHalf;
        char phrase[MAXSIZE];

   printf("Please enter a phrase to find out if it is palindrome or not!\n");
        gets(phrase);
        firstHalf = CreateStack();
        IsPalindrome(phrase, firstHalf);

        return 0;
}
```

```
//stackDynamic.c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include "stackDynamic.h"

#define MAXSIZE 50

struct Node{
	char c;
	struct Node *next;
};
typedef struct Node StackRecord;
typedef StackRecord *Stack;

void IsPalindrome(char phrase[MAXSIZE], Stack firstHalf){
	char ordered[MAXSIZE]; // I will make the all the phrase's letters bigger and put them in this char array.
	char firstOrdered[MAXSIZE/2], secondOrdered[MAXSIZE/2];
	Stack otherHalf, temp;
	int i = 0, j = 0, counter = 0;
	otherHalf = CreateStack(); // this will be the second half of the phrase, I will use temporary stack to reverse it and compare with s

	while(phrase[i] != '\0'){
		if (phrase[i] >= 65 && phrase[i] <= 90){
			ordered[j] = phrase[i];
```

```
                j++;
                counter++;
        }
        else if (phrase[i] >= 97 && phrase[i] <= 122) {
                ordered[j] = toupper(phrase[i]);
                j++;
                counter++;// this counter for the length of the array.
        }

        i++;
}
ordered[j] = '\0'; // now I created my array with all the only letters with all upper letters.


for(i=0; i<counter/2 ;i++) // this for loop for the first half.
        PushStack(ordered[i], firstHalf);

for(i=counter/2; i<counter ;i++) // this for loop for the second half.
        PushStack(ordered[i], otherHalf);

temp = CreateStack(); // this stack is temporary, I will pop from the otherHalf and push to temp so that I can have the same order.

while(!IsEmptyStack(otherHalf)) // push from the other half to temp
        PushStack(PopStack(otherHalf), temp);

if (counter % 2 == 1)
        PopStack(temp); // this pop operation for getting rid of the middle element if the phrase is a odd number.
```

```
	for(i=0; i<counter/2 ;i++) // I popped from first stack and send it to firstOrdered array.
		firstOrdered[i] = PopStack(firstHalf);

	j = 0;
	for(i=counter/2; i<counter ;i++){ // I popped from temp stack and send it to secondOrdered array.
		secondOrdered[j] = PopStack(temp);
		j++;
	}

	int flag = 1; // this is a flag for comparision two arrays
	for(i=0; i<counter/2 ;i++){
		if(firstOrdered[i] != secondOrdered[i]){
			flag = 0;
			break;
		}
	}

	if(flag == 1)
		printf("The phrase is palindrome!\n");
	else
		printf("The phrase is not palindrome!\n");
}

Stack CreateStack(){
	Stack s;

	s = (struct Node*)malloc(sizeof(struct Node));
	if (s == NULL){
```

```
		printf("Out of memory!\n");
		exit(-1);
	}

	s->next = NULL;

	return s;
}

void PushStack(char x, Stack s){
	Stack temp;
	temp = (struct Node*)malloc(sizeof(struct Node));
	if (temp == NULL){
		printf("Out of memory!\n");
		exit(-1);
	}

	temp->c = x;
	temp->next = s->next;
	s->next = temp;
}

int PopStack(Stack s){
	if(!IsEmptyStack(s)){
		Stack removal;
		char poppingVal;
		removal = s->next;
		s->next = s->next->next;
		poppingVal = removal->c;
```

```
                free(removal);
                return poppingVal;
        }
}

int IsEmptyStack(Stack s){
                if (s->next == NULL)
                return 1;
        else
                return 0;
}
```

```
//stackDynamic.h
typedef struct Node StackRecord;
typedef StackRecord *Stack;

void IsPalindrome(char *, Stack s);
Stack CreateStack();
void PushStack(char , Stack s);
int PopStack(Stack);
int IsEmptyStack(Stack);
```