**Question 1: SLL() and Palindrome() functions in LinkedLists**

**C Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct Node {

int data;

struct Node\* next;

};

void sll\_insert(struct Node\*\* head, int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = \*head;

\*head = newNode;

}

bool isPalindrome(struct Node\* head) {

if (head == NULL)

return true;

struct Node\* slow = head;

struct Node\* fast = head;

struct Node\* stack = NULL;

while (fast != NULL && fast->next != NULL) {

sll\_insert(&stack, slow->data);

slow = slow->next;

fast = fast->next->next;

}

if (fast != NULL)

slow = slow->next;

while (slow != NULL) {

if (slow->data != stack->data)

return false;

slow = slow->next;

stack = stack->next;

}

return true;

}

void printList(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

printf("%d -> ", current->data);

current = current->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

sll\_insert(&head, 1);

sll\_insert(&head, 2);

sll\_insert(&head, 3);

sll\_insert(&head, 2);

sll\_insert(&head, 1);

printf("Linked List: ");

printList(head);

if (isPalindrome(head))

printf("The linked list is a palindrome.\n");

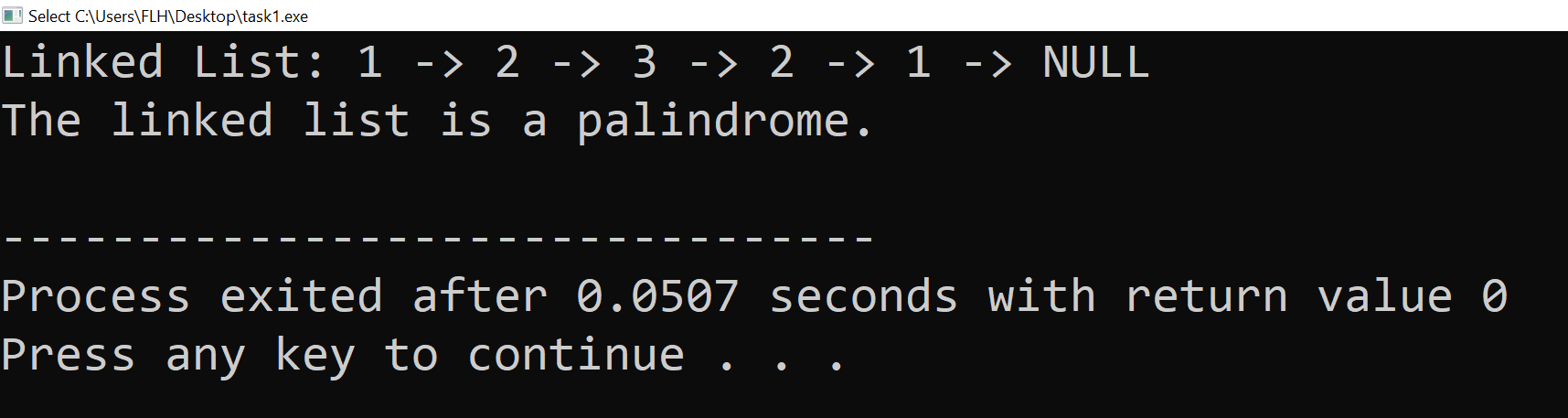
else

printf("The linked list is not a palindrome.\n");

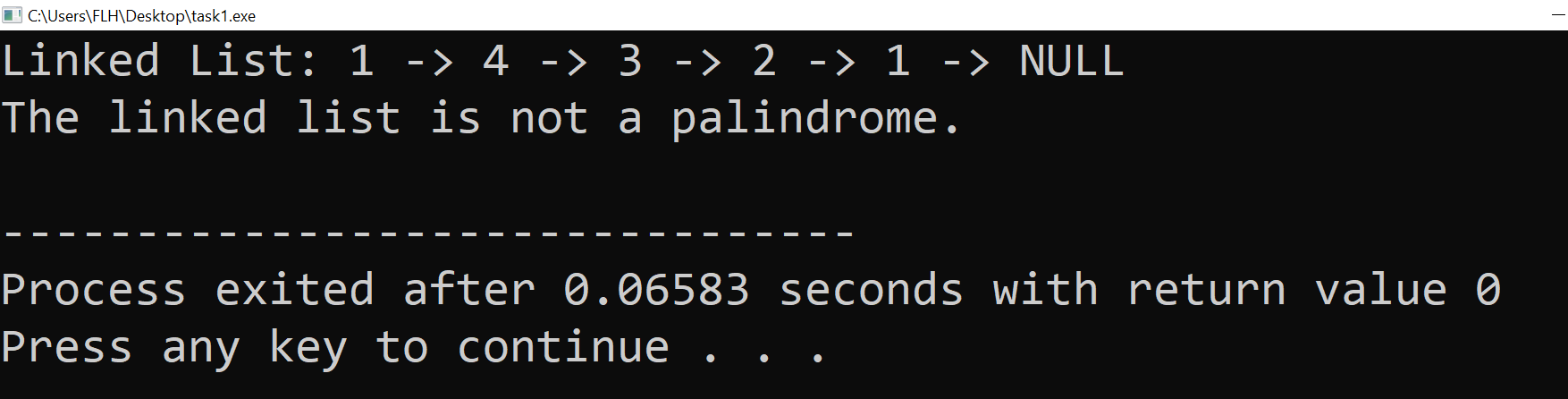
return 0;

}

**Ouput ScreenShot: (Case 1)**

****

**Ouput ScreenShot: (Case 2)**

****

**Question 2: Stacks using Arrays**

**C Code:**

#include<stdio.h>

#include<stdlib.h>

struct stack{

int top;

int size;

int \*arr;

};

int isFull(struct stack \*s){

if(s->top==s->size-1){

return 1;

}

else{

return 0;

}

}

int isEmpty(struct stack \*s){

if(s->top==-1){

return 1;

}

else{

return 0;

}

}

void push(struct stack \*s, int val){

if(isFull(s)){

printf("Stack is Full, Cannot Push anything into Stack\n");

}

else{

s->top++;

s->arr[s->top]=val;

printf("%d has been successfully pushed into stack\n", val);

}

}

int pop(struct stack \*s){

int val=-1;

if(isEmpty(s)){

printf("Stack Underflow, Cannot Pop from Stack\n");

}

else{

int val=s->arr[s->top];

s->top--;

return val;

}

}

int peek(struct stack \*s, int pos){

int arrayInd=s->top-pos +1;

if(arrayInd<0){

printf("Invalid Position\n");

return -1;

}

else{

return s->arr[arrayInd];

}

}

int main(){

struct stack \*s=(struct stack\*)malloc(sizeof(struct stack));

s->size=5;

s->top=-1;

s->arr=(int\*)malloc(s->size\*sizeof(int));

push(s, 3);

push(s, 5);

push(s, 7);

push(s, 9);

push(s, 11);

push(s, 13); // Stack Overflown, will not be pushed

push(s, 15);

printf("%d has been popped out\n", pop(s)); // 2 elements are popped now

printf("%d has been popped out\n", pop(s));

push(s, 13); // Now they'll be pushed

push(s, 15);

int i;

for(i=1; i<=s->size; i++){

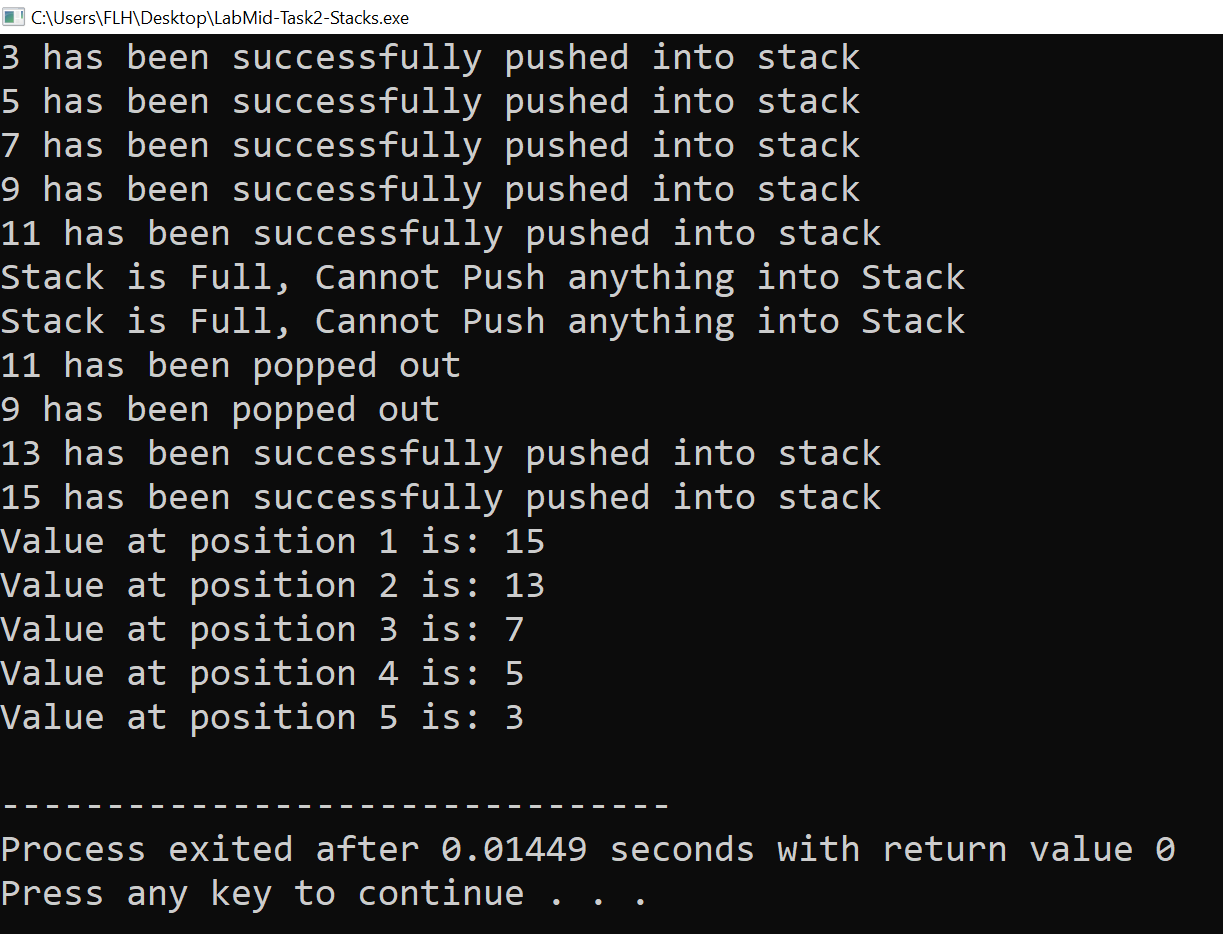
printf("Value at position %d is: %d\n", i, peek(s, i));

}

return 0;

}

**Output ScreenShot:**

****