```c
// C program to reverse a string using stack
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// A structure to represent a stack
struct Stack {
    int top;
    unsigned capacity;
    char* array;
};

// function to create a stack of given
// capacity. It initializes size of stack as 0
struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack
        = (struct Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array
        = (char*)malloc(stack->capacity * sizeof(char));
    return stack;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
    return stack->top == stack->capacity - 1;
}

// Stack is empty when top is equal to -1
int isEmpty(struct Stack* stack)
{
    return stack->top == -1;
}

// Function to add an item to stack.
```

```
40: // It increases top by 1
41: void push(struct Stack* stack, char item)
42: {
43:     if (isFull(stack))
44:         return;
45:     stack->array[++stack->top] = item;
46: }
47:
48: // Function to remove an item from stack.
49: // It decreases top by 1
50: char pop(struct Stack* stack)
51: {
52:     if (isEmpty(stack))
53:         return INT_MIN;
54:     return stack->array[stack->top--];
55: }
56:
57: // A stack based function to reverse a string
58: void reverse(char str[])
59: {
60:     // Create a stack of capacity
61:     // equal to length of string
62:     int n = strlen(str);
63:     struct Stack* stack = createStack(n);
64:
65:     // Push all characters of string to stack
66:     int i;
67:     for (i = 0; i < n; i++)
68:         push(stack, str[i]);
69:
70:     // Pop all characters of string and
71:     // put them back to str
72:     for (i = 0; i < n; i++)
73:         str[i] = pop(stack);
74: }
75:
76: // Driver program to test above functions
77: int main()
78: {
```

```
79:     char str[] = "Vivekkaspa";
80:
81:     reverse(str);
82:     printf("Reversed string is %s", str);
83:
84:     return 0;
85: }
86:
```