Stack Using Dynamic Memory Allocation (Using Linked List.)

```c
/*

Que : Write a C program for Creation of Stack Using Linked List.
Owner: Rushikesh Sanjay Pokharkar
Batch: PPA9

*/

//                                        *********  Solution  *********


#include<stdio.h> //Include Necessary Header Files.
#include<stdlib.h>

struct STACK // Created a structure of a Stack.
{
	int data;
	struct STACK* next; // next pointer represents the top value.
};


int isEmpty(struct STACK** stackptr) // isEmpty function to check the stack is empty or
not.
{
	return (*stackptr == NULL);
}

struct STACK* createNode() // createNode function for- creation of node which is going to
insert at top.
{
	struct STACK* newnode = NULL;

	newnode = (struct STACK*)malloc(sizeof(struct STACK)); // memory allocated using
DMA.

	if (newnode == NULL) // Check for stack is full or not.
	{
		printf("Stack is FULL You can not perform PUSH Operation on it.\n");
	}
	else
	{
		int data;
		printf("Enter a data: ");
		scanf_s("%d", &data);

		newnode->data = data;
		newnode->next = NULL;
	}
	return newnode; // Return created node to push function.
```

```
}

void push(struct STACK** stackptr) // Push function to insert new datat at the top of the
stack.
{
	struct STACK* newnode = NULL;
	newnode = createNode();

	// Linking of the top and new inserted node.
	newnode->next = *stackptr;
	*stackptr = newnode;
}

int pop(struct STACK** stackptr) // pop function to delete the top most data value from
stack.
{
	struct STACK* deletenode = NULL;
	int data = (*stackptr)->data;

	// Linking of top most node(deletenode) and his next node.
	deletenode = *stackptr;
	*stackptr = (*stackptr)->next;

	free(deletenode); // Free the dynamic memory given to the data.
	return data; // Return the removed data to the function call of pop function.
}

void main()
{
	int choice;
	struct STACK* stackptr = NULL;

	do
	{
		// Menue of the stack operations.
		printf("Enter Your Choice: \n");
		printf("0. Exit.\n");
		printf("1. Push.\n");
		printf("2. Pop.\n");
		printf("Choice = ");
		scanf_s("%d", &choice);

		switch (choice)
		{
		case 0:
			printf("Thank You!!!\n");
			break;
		case 1:
			push(&stackptr); // Function to add data into the stack.
			break;
		case 2:
			if (isEmpty(&stackptr)) // Function in condition to check if the
stack is empty or not.
			{
				printf("Stack is Empty You can not perform POP Operation on
it.\n");
			}
			else
```

```
                        {
                                printf("The value %d is Poped From Stack.\n", pop(&stackptr));
// Function call to delete data from stack.
                        }
                        break;
                default:
                        printf("Please Enter a Valid Choice.\n");
                }
        } while (choice != 0);
}
```