Lab Assignment-7

ROLL: 2005535 | NAME: SAHIL SINGH | DATE: 07/09/21

QUES 1: [A] Write a menu driven program to perform the following operations in a STACK ADT (Using an Array) by using suitable user defined functions for each case.

1. PUSH an element into the stack [Define Isfull() function to check overflow]

2. POP an element from the stack [Define Isempty() function to check underflow]

3. Display a stack

4. Copy the content of one stack into another (Without using any additional data structure)

5. Sort the elements of the stack (Without using any additional data structure)

SOLUTION:

```c
#include <stdio.h>
#define MAX 100

typedef struct Stack
{
    int topCount;
    int data[MAX];
} Stack;

int isFull(Stack *);
int isEmpty(Stack *);
void push(Stack *, int);
int pop(Stack *);
void display(Stack *);
void copy(Stack *, Stack *);
void sorted_insert(Stack *, int);
void sort(Stack *);

int main()
{
    Stack stack = {-1};
    Stack stack2 = {-1};
    int choice;
    do
    {
        printf("1) Insert a number\n2) Delete top element\n");
        printf("3) display stack\n4) Copy stack\n5) Sort stack\n6) Exit\n->: ");
        scanf("%d", &choice);
        int val;
        switch (choice)
        {
        case 1:
            printf("Enter a value to insert: ");
            scanf("%d", &val);
            push(&stack, val);
            break;
        case 2:
            printf("%d was deleted!\n", pop(&stack));
            break;
```

```
        case 3:
            printf("\ntop->");
            display(&stack);
            break;
        case 4:
            printf("\ntop->");
            copy(&stack2, &stack);
            display(&stack2);
            break;
        case 5:
            printf("\ntop->");
            sort(&stack);
            display(&stack);
            break;
        default:
            printf("\nExiting...\n");
        }
        printf("--------------------------------\n");
    } while (choice >= 1 && choice <= 5);
    return 0;
}
void push(Stack *stack, int num)
{
    if (isFull(stack))
    {
        printf("Overflow!\n");
        return;
    }
    stack->data[++stack->topCount] = num;
}
int pop(Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Underflow!\n");
        return -99999;
    }
    return stack->data[stack->topCount--];
}
void display(Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("\b\b \n");
        return;
    }
    int temp = pop(stack);
    printf("%d->", temp);
    display(stack);
    push(stack, temp);
}
int isFull(Stack *stack)
{
    if (stack->topCount == MAX - 1)
        return 1;
    return 0;
```

```
}
int isEmpty(Stack *stack)
{
    if (stack->topCount == -1)
        return 1;
    return 0;
}
void copy(Stack *dest, Stack *src)
{
    if (isEmpty(src))
    {
        dest->topCount = -1;
        return;
    }
    pop(src);
    copy(dest, src);
    push(dest, src->data[src->topCount + 1]);
    push(src, src->data[src->topCount + 1]);
}
void sorted_insert(Stack *stack, int val)
{
    if (isEmpty(stack) || val > stack->topCount)
    {
        push(stack, val);
        return;
    }
    int temp = pop(stack);
    sorted_insert(stack, temp);
    push(stack, temp);
}
void sort(Stack *stack)
{
    if (isEmpty(stack))
        return;
    int temp = pop(stack);
    sort(stack);
    sorted_insert(stack, temp);
}
```

OUTPUT:

```
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 1
Enter a value to insert: 12
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
```

```
->: 1
Enter a value to insert: 13
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 1
Enter a value to insert: 14
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 3

top->14->13->12 >
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 2
14 was deleted!
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 3

top->13->12 >
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 4

top->13->12 >
------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
```

```
6) Exit
->: 1
Enter a value to insert: 5
--------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 3

top->5->13->12 >
--------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 1
Enter a value to insert: 56
--------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 5

top->56->5->13->12 >
--------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 3

top->56->5->13->12 >
--------------------------------
1) Insert a number
2) Delete top element
3) display stack
4) Copy stack
5) Sort stack
6) Exit
->: 6

Exiting...
--------------------------------
```

QUES 2: [B] Write a menu driven program to perform the following operations in a STACK ADT (Using linked list) by using suitable user defined functions for each case.

1. PUSH an element into the stack

2. POP an element from the stack [Define Isempty() function to check underflow]

3. Display a stack

SOLUTION:

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Stack
{
    int data;
    struct Stack *link;
} Stack;

void push(Stack **, int);
int pop(Stack **);
int isEmpty(Stack *);
void display(Stack *);

int main()
{
    Stack *stack = NULL;
    int choice;
    do
    {
        int val;
        printf("1) Insert in stack\n2) Display\n3) Delete top\n4) Exit\n->: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter value: ");
            scanf("%d", &val);
            push(&stack, val);
            break;
        case 2:
            printf("\ntop->");
            display(stack);
            break;
        case 3:
            printf("\nDeleted element: %d\n", pop(&stack));
            break;
        default:
            printf("\nExiting...\n");
        }
        printf("----------------------------\n");
    } while (choice >= 1 && choice <= 3);
    return 0;
}
void push(Stack **stack_top, int num)
{
    Stack *temp = (Stack *)malloc(sizeof(Stack));
```

```
    temp->data = num;
    temp->link = *stack_top;
    *stack_top = temp;
}
int pop(Stack **stack_top)
{
    if (isEmpty(*stack_top))
    {
        printf("\nUnderflow!");
        return -9999999;
    }
    Stack *temp = (*stack_top);
    *stack_top = (*stack_top)->link;
    int val = temp->data;
    free(temp);
    return val;
}
int isEmpty(Stack *stack_top)
{
    if (!stack_top)
        return 1;
    return 0;
}
void display(Stack *stack_top)
{
    if (isEmpty(stack_top))
    {
        printf("\b\b \n");
        return;
    }
    int temp = pop(&stack_top);
    printf("%d->", temp);
    display(stack_top);
    push(&stack_top, temp);
}
```

OUTPUT:

```
1) Insert in stack
2) Display
3) Delete top
4) Exit
->: 1
Enter value: 10
----------------------------
1) Insert in stack
2) Display
3) Delete top
4) Exit
->: 1
Enter value: 20
----------------------------
1) Insert in stack
2) Display
3) Delete top
4) Exit
```

```
->: 1
Enter value: 30
----------------------------
1) Insert in stack
2) Display
3) Delete top
4) Exit
->: 2

top->30->20->10 >
----------------------------
1) Insert in stack
2) Display
3) Delete top
4) Exit
->: 3

Deleted element: 30
----------------------------
1) Insert in stack
2) Display
3) Delete top
4) Exit
->: 2

top->20->10 >
----------------------------
1) Insert in stack
2) Display
3) Delete top
4) Exit
->: 4

Exiting...
----------------------------
```

QUES 3: [C] Write a program to implement two numbers of stacks in one array to minimize overflow in any one of the stack.

SOLUTION:

```c
#include <stdio.h>
#define MAX 100
typedef struct Stack
{
    int top1;
    int top2;
    int data[MAX];
} Stack;
int isFull(Stack *);
//For first stack
int isEmpty1(Stack *);
void push1(Stack *, int);
int pop1(Stack *);
void display1(Stack *);
//For second stack
```

```c
int isEmpty2(Stack *);
void push2(Stack *, int);
int pop2(Stack *);
void display2(Stack *);
int main()
{
    Stack stack = {-1, MAX};
    int choice;
    do
    {
        printf("Enter stack ID 1 for first stack and 2 for second!\n");
        printf("1) Insert\n2) Display\n3) Delete\n4) Exit\n->: ");
        scanf("%d", &choice);
        int val, id;
        switch (choice)
        {
        case 1:
            printf("Stack ID: ");
            scanf("%d", &id);
            printf("Enter value: ");
            scanf("%d", &val);
            (id - 1) ? push2(&stack, val)
                     : push1(&stack, val);
            break;
        case 2:
            printf("Stack ID: ");
            scanf("%d", &id);
            (id - 1) ? display2(&stack) : display1(&stack);
            break;
        case 3:
            printf("Stack ID: ");
            scanf("%d", &id);
            (id - 1) ? pop2(&stack) : pop1(&stack);
            (id - 1) ? display2(&stack) : display1(&stack);
            break;
        default:
            printf("\bExiting...\n");
        }
        printf("----------------------------\n");
    } while (choice >= 1 && choice <= 3);
    return 0;
}
int isFull(Stack *stack)
{
    if (stack->top1 + 1 == stack->top2)
        return 1;
    return 0;
}
int isEmpty1(Stack *stack)
{
    if (stack->top1 == -1)
        return 1;
    return 0;
}
void push1(Stack *stack, int num)
{
```

```
    if (isFull(stack))
    {
        printf("Overflow!\n");
        return;
    }
    stack->data[++stack->top1] = num;
}
int pop1(Stack *stack)
{
    if (isEmpty1(stack))
    {
        printf("Underflow!\n");
        return -99999;
    }
    return stack->data[stack->top1--];
}
void display1(Stack *stack)
{
    if (isEmpty1(stack))
    {
        printf("\b\b \n");
        return;
    }
    int temp = pop1(stack);
    printf("%d->", temp);
    display1(stack);
    push1(stack, temp);
}
////////////////////////////////////////////////////////////////////////////////
int isEmpty2(Stack *stack)
{
    if (stack->top2 == MAX)
        return 1;
    return 0;
}
void push2(Stack *stack, int num)
{
    if (isFull(stack))
    {
        printf("Overflow!\n");
        return;
    }
    stack->data[--stack->top2] = num;
}
int pop2(Stack *stack)
{
    if (isEmpty2(stack))
    {
        printf("Underflow!\n");
        return -99999;
    }
    return stack->data[stack->top2++];
}
void display2(Stack *stack)
{
    if (isEmpty2(stack))
```

```
    {
        printf("\b\b \n");
        return;
    }
    int temp = pop2(stack);
    printf("%d->", temp);
    display2(stack);
    push2(stack, temp);
}
```

OUTPUT:

```
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 1
Stack ID: 1
Enter value: 1
---------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 1
Stack ID: 1
Enter value: 2
---------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 1
Stack ID: 1
Enter value: 3
---------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 2
Stack ID: 1
3->2->1 >
---------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 1
Stack ID: 2
Enter value: 23
```

```
--------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 1
Stack ID: 2
Enter value: 24
--------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 1
Stack ID: 2
Enter value: 25
--------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 3
Stack ID: 1
2->1 >
--------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 2
Stack ID: 2
25->24->23 >
--------------------------
Enter stack ID 1 for first stack and 2 for second!
1) Insert
2) Display
3) Delete
4) Exit
->: 4
Exiting...
--------------------------
```