Computer Engineering Department, S.V.N.I.T. Surat.
B Tech (CO) –II[nd] Year semester-III
Course: *Data Structures CO203*

Assignment-V

1. Write a program to implement a stack and perform basic operations of stack.

1) push

2) pop

3) peek

4) isfull

5) isempty

A.) Code [for "int" Data Type]:

```c
// Write a program to implement a stack and perform basic operations of stack.
// 1) push
// 2) pop
// 3) peek
// 4) isfull
// 5) isempty
// 6.) To Display whole Stack

#include <stdio.h>

#define MAX 1000 //Maximum number of elements that can be stored

int top = -1, stack[MAX];

int isfull()
{
    if (top == MAX - 1)
        return 1;
    else
        return 0;
}

int isempty()
{
    if (top == -1)
        return 1;
    else
        return 0;
}
```

```c
void push()
{
    int val;

    if (isfull())
    {
        printf("\nStack is Full!\n");
    }
    else
    {
        printf("\nEnter element to push : ");
        scanf("%d", &val);
        top = top + 1;
        stack[top] = val;
    }
}

void pop()
{
    if (isempty())
    {
        printf("\nStack is Empty!\n");
    }
    else
    {
        printf("\nDeleted element is %d\n", stack[top]);
        top = top - 1;
    }
}

void peek()
{
    if (isempty())
    {
        printf("\nStack is Empty!\n");
    }
    else
    {
        printf("\nStack Top : %d\n", stack[top]);
    }
}

void display()
{
    int i;

    if (isempty())
    {
        printf("\nStack is Empty!\n");
    }
```

```
    else
    {
        printf("\nStack :\n");
        for (i = top; i >= 0; --i)
            printf("%d\n", stack[i]);
    }
}

int main()
{
    int choice;
    printf("\nStack Operation\n");
    printf("1 - > Push\n");
    printf("2 - > Pop\n");
    printf("3 - > Peek Top\n");
    printf("4 - > Check Stack is Full\n");
    printf("5 - > Check Stack is Empty\n");
    printf("6 - > Display Whole Stack\n");
    printf("7 - > Exit\n");

    while (1) //infinite loop, will end when choice will be 7
    {
        printf("\nEnter your choice [1-7] : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            peek();
            break;
        case 4:
            if (isfull())
                printf("\nStack is Full!\n");
            else
                printf("\nStack is Not Full!\n");
            break;
        case 5:
            if (isempty())
                printf("\nStack is Empty!\n");
            else
                printf("\nStack is Not Empty!\n");
            break;
        case 6:
            display();
```

```
            break;
        case 7:
            return 0;

        default:
            printf("\nWrong Choice Entered!");
        }
    }

    return 0;
}
```

Test Cases:

```
Stack Operation
1 - > Push
2 - > Pop
3 - > Peek Top
4 - > Check Stack is Full
5 - > Check Stack is Empty
6 - > Display Whole Stack
7 - > Exit

Enter your choice [1-7] : 5

Stack is Empty!

Enter your choice [1-7] : 1

Enter element to push : 10

Enter your choice [1-7] : 1

Enter element to push : 12

Enter your choice [1-7] : 1

Enter element to push : 15

Enter your choice [1-7] : 6

Stack :
15
12
10
```

```
Enter your choice [1-7] : 6

Stack :
15
12
10

Enter your choice [1-7] : 2

Deleted element is 15

Enter your choice [1-7] : 6

Stack :
12
10

Enter your choice [1-7] : 3

Stack Top : 12

Enter your choice [1-7] : 4

Stack is Not Full!

Enter your choice [1-7] : 5

Stack is Not Empty!

Enter your choice [1-7] : 7
```

B.) Code [for “char” Data Type]:

```c
// Write a program to implement a stack and perform basic operations of stack.
// 1) push
// 2) pop
// 3) peek
// 4) isfull
// 5) isempty
// 6.) To Display whole Stack

#include <stdio.h>
```

```c
#define MAX 1000 //Maximum number of elements that can be stored

int top = -1;
char stack[MAX];

int isfull()
{
    if (top == MAX - 1)
        return 1;
    else
        return 0;
}

int isempty()
{
    if (top == -1)
        return 1;
    else
        return 0;
}

void push()
{
    char val;

    if (isfull())
    {
        printf("\nStack is Full!\n");
    }
    else
    {
        fflush(stdin);
        printf("\nEnter char to push : ");
        scanf("%c", &val);
        top = top + 1;
        stack[top] = val;
    }
}

void pop()
{
    if (isempty())
    {
        printf("\nStack is Empty!\n");
    }
    else
    {
        printf("\nDeleted element is %c\n", stack[top]);
        top = top - 1;
    }
```

```
}

void peek()
{
    if (isempty())
    {
        printf("\nStack is Empty!\n");
    }
    else
    {
        printf("\nStack Top : %c\n", stack[top]);
    }
}

void display()
{
    int i;

    if (isempty())
    {
        printf("\nStack is Empty!\n");
    }
    else
    {
        printf("\nStack :\n");
        for (i = top; i >= 0; --i)
            printf("%c\n", stack[i]);
    }
}

int main()
{
    int choice;
    printf("\nStack Operation\n");
    printf("1 - > Push\n");
    printf("2 - > Pop\n");
    printf("3 - > Peek Top\n");
    printf("4 - > Check Stack is Full\n");
    printf("5 - > Check Stack is Empty\n");
    printf("6 - > Display Whole Stack\n");
    printf("7 - > Exit\n");

    while (1) //infinite loop, will end when choice will be 7
    {
        printf("\nEnter your choice [1-7] : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
```

```
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            peek();
            break;
        case 4:
            if (isfull())
                printf("\nStack is Full!\n");
            else
                printf("\nStack is Not Full!\n");
            break;
        case 5:
            if (isempty())
                printf("\nStack is Empty!\n");
            else
                printf("\nStack is Not Empty!\n");
            break;
        case 6:
            display();
            break;
        case 7:
            return 0;

        default:
            printf("\nWrong Choice Entered!");
        }
    }

    return 0;
}
```

Test Cases:

```
Stack Operation
1 - > Push
2 - > Pop
3 - > Peek Top
4 - > Check Stack is Full
5 - > Check Stack is Empty
6 - > Display Whole Stack
7 - > Exit

Enter your choice [1-7] : 5

Stack is Empty!

Enter your choice [1-7] : 1

Enter char to push : a

Enter your choice [1-7] : 1

Enter char to push : b

Enter your choice [1-7] : 1

Enter char to push : c

Enter your choice [1-7] : 6

Stack :
c
b
a
```

```
Enter your choice [1-7] : 6

Stack :
c
b
a

Enter your choice [1-7] : 2

Deleted element is c

Enter your choice [1-7] : 6

Stack :
b
a

Enter your choice [1-7] : 3

Stack Top : b

Enter your choice [1-7] : 4

Stack is Not Full!

Enter your choice [1-7] : 5

Stack is Not Empty!

Enter your choice [1-7] : 7
```

2. Write a program to check string is palindrome using stack.

Code:

```c
//Write a program to check string is palindrome using stack.
#include <stdio.h>
#include <string.h>

#define MAX 1000 //Maximum number of elements that can be stored

int top = -1;
char stack[MAX];

int isfull()
{
    if (top == MAX - 1)
        return 1;
    else
        return 0;
}

int isempty()
{
    if (top == -1)
        return 1;
    else
        return 0;
}

void push(char letter)
{
    if (isfull())
    {
        // printf("\nStack is Full!\n");
    }
    else
    {
        top = top + 1;
        stack[top] = letter;
    }
}

char pop()
{
    if (isempty())
    {
        // printf("\nStack is Empty!\n");
        return '0';
    }
    else
```

```
    {
        char tp = stack[top];
        top = top - 1;
        return tp;
    }
}
// ~~~~~~~~ ALGORITHM ~~~~~~~~~~~~~~
// len(s1) = returns length of string s1

// 1.) stack.top = -1
// 2.) Read s1
// 3.) len1 = len(s1)

// 4.) i=0;
// 5.) Repeat Step 6 & 7 while i < (len1)/2
// 6.) push(s1[i], stack)
// 7.) i = i + 1

// 8.) if len is odd [len%2!=0]
// 9.)    i = i+1

// 10.) Repeat Step 11 & 12 while not isempty(stack)
// 11.)  if(stack.top != s[i]) return false;
// 12.)  i = i + 1; pop(stack.top);

// 13.) return true

int check_palindrome(char *str)
{
    int len = strlen(str);
    int i, mid = len / 2;
    for (i = 0; i < mid; i++)
    {
        push(str[i]);
    }

    if (len % 2 == 1) // Odd Length String
        i += 1;

    while (i < len)
    {
        char ele = pop();
        if (ele != (char)str[i])
        {
            return 0; // Cant Be Palindrome
        }
        i += 1;
    }

    top = -1; //Empty the Stack
```

```
    return 1; // Palindrome
}

int main()
{
    int t;
    printf("Enter the Number of Strings to Check for Palindrome:\n");

    scanf("%d", &t);

    while (t--)
    {

        char str[MAX];

        printf("Enter String :\n");
        fflush(stdin);

        gets(str);

        if (check_palindrome(str) != 0)
        {
            printf("%s : Palidromic String\n", str);
        }
        else
        {
            printf("%s : Not Palindromic String\n", str);
        }
    }

    return 0;
}
```

Test Cases:

```
Enter the Number of Strings to Check for Palindrome:
10
Enter String :
civic
civic : Palidromic String
Enter String :
apple
apple : Not Palindromic String
Enter String :
level
level : Palidromic String
Enter String :
Radar
Radar : Not Palindromic String
Enter String :
RefeR
RefeR : Palidromic String
Enter String :
Palidrome
Palidrome : Not Palindromic String
Enter String :
redivider
redivider : Palidromic String
Enter String :
racecar
racecar : Palidromic String
Enter String :
rotor
rotor : Palidromic String
Enter String :
datastructure
datastructure : Not Palindromic String
```

3. Write a program to sort the string using stack.

Code:

```c
// Write a program to sort the string using stack
#include <stdio.h>

#define MAX 1000 //Maximum number of elements that can be stored

//To Avoid the Conflicts of Top of "stack1" and "temp" stack
// I will be Using Struct
struct stack
{
    char stk[MAX];
    int top;

} stack1, temp;

char peek(struct stack *s)
{
    if (s->top == -1)
    {
        printf("stack is empty");
        return '$';
    }
    //Return the Character at the Top of Stack
    return s->stk[s->top];
}

void push(struct stack *s, char c)
{
    if (s->top == MAX - 1)
    {
        printf("stack overflow");
        return;
    }
    s->top += 1;
    s->stk[s->top] = c;
}

char pop(struct stack *s)
{
    if (s->top == -1)
    {
        printf("stack underflow");
        return '$';
    }
    // return the Pop Character and then Decrements the Top
    return s->stk[s->top--];
    // Post Decrement
```

```
}

int main()
{
    char str[MAX];

    // Initialise Two Empty Stack
    stack1.top = -1;
    temp.top = -1;

    printf("Enter the String to Sort :\n");

    fgets(str, MAX, stdin);

    // Push the First Char of String in Stack "stack1"
    push(&stack1, str[0]);

    int i = 0;

    // Iterate till End of String
    for (i = 1; str[i] != '\0'; i++)
    {
        // If ASCII of Current Character is Less than Top
        // Push it in stack1
        if (peek(&stack1) >= str[i])
        {
            push(&stack1, str[i]);
        }
        else
        {
            // Otherwise Until the Stack1 is Not Empty
            while (stack1.top != -1)
            {
                // If top [ASCII] is less than ith char of str [ASCII]
                if (peek(&stack1) < str[i])
                {
                    //Push the Element from Stack to temp
                    temp.top += 1;
                    temp.stk[temp.top] = pop(&stack1);
                }
                else
                {
                    // Otherwise Break
                    break;
                }
            }

            push(&stack1, str[i]);

            while (temp.top != -1)
```

```
            {
                // Now Transfer all Character from "temp" to "stack1"
                push(&stack1, pop(&temp));
            }
        }
    }

    char str_sort[MAX];

    int j = 0;

    while (stack1.top != -1)
    {
        str_sort[j] = pop(&stack1);
        j += 1;
    }

    str_sort[j] = '\0'; // Append /0 at End

    printf("Sorted String : %s\n", str_sort);

    return 0;
}
```

Test Cases:

```
Enter the String to Sort :
bbccdefbzzpqrste
Sorted String :
bbbccdeefpqrstzz
```

```
Enter the String to Sort :
323534654
Sorted String :
233344556
```

```
Enter the String to Sort :
ZXGADzxfre5231
Sorted String :
1235ADGXZefrxz
```

Submitted By:
Roll Number: **U19CS012** (*D-12*)
Name: *Bhagya Rana*