WEEK 3

Lab Exercises:

Write a 'C' program to:

1: Implement a menu driven program to define a stack of characters. Include push, pop and display functions. Also include functions for checking error conditions such as underflow and overflow (ref. Figure 1) by defining isEmpty and isFull functions. Use these function in push, pop and display functions appropriately. Use type defined structure to define a STACK containing a character array and an integer top. Do not use global variables.

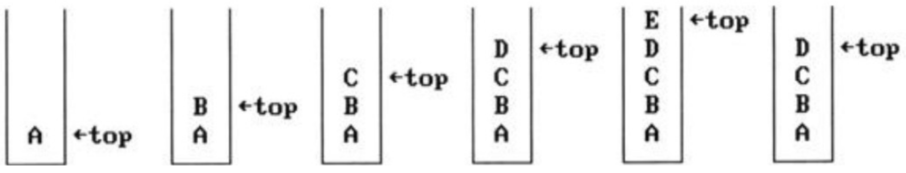


Figure 1: Inserting and deleting elements in a stack

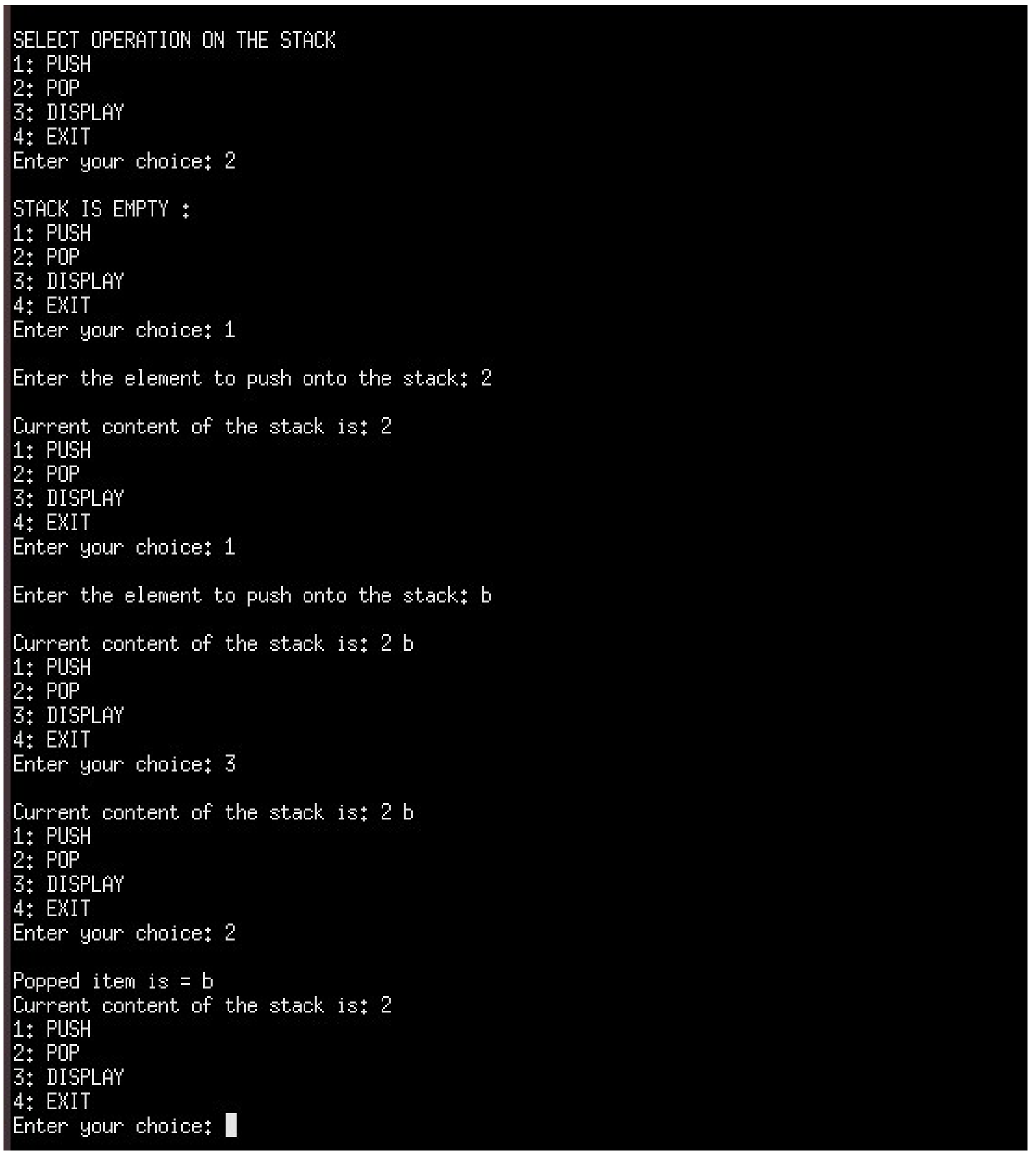
Code:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
#define UNDERFLOW '\0'
typedef enum
{
NO = 0,
YES = 1,
} BOOL;
BOOL isStackFull (int tos)
{
if (tos == SIZE - 1)
return YES;
return NO;
}
BOOL isStackEmpty (int tos)
{
if (tos == -1)
return YES;
return NO;
}
void push (char *stack, int item, int *tos)
{
```

```
if (isStackFull (*tos))
{
printf("\nSTACK IS FULL\n");
return;
}
(*tos) += 1;
*(stack + (*tos)) = item;
}
char pop (char *stack, int *tos)
{
if (isStackEmpty (*tos))
{
printf("\nSTACK IS EMPTY : ");
return UNDERFLOW;
}return *(stack + ((*tos)--));
}
void display(char *stack,int tos)
{
char *pi;
for(pi=stack; pi<=stack+tos; ++pi)
printf("%c ",*pi);
}
int main(int argc, const char * argv[])
{
int tos = -1;
char *stack = (char *)malloc(sizeof(char *));
char ch = '3';
printf("\nSELECT OPERATION ON THE STACK");
while (ch == '1' || ch == '2' || ch == '3')
{
printf("\n1: PUSH");
printf("\n2: POP");
printf("\n3: DISPLAY");
printf("\n4: EXIT");
printf("\nEnter your choice: ");
scanf(" %c", &ch);
if (ch == '1')
{
char item;
printf("\nEnter the element to push onto the stack: ");
scanf(" %c", &item);
push(stack, item, &tos);
if (!isStackFull(tos))
{
printf("\nCurrent content of the stack is: ");
display(stack, tos);
}
}
else if (ch == '2')
{
```

```
char item = pop(stack, &tos);
if (item != UNDERFLOW)
{
printf("\nPopped item is = %c", item);
printf("\nCurrent content of the stack is: ");
display(stack, tos);
}}
else if (ch == '3')
{
printf("\nCurrent content of the stack is: ");
display(stack, tos);
}
else
exit(0);
}
return 0;
}
```

Test Case:



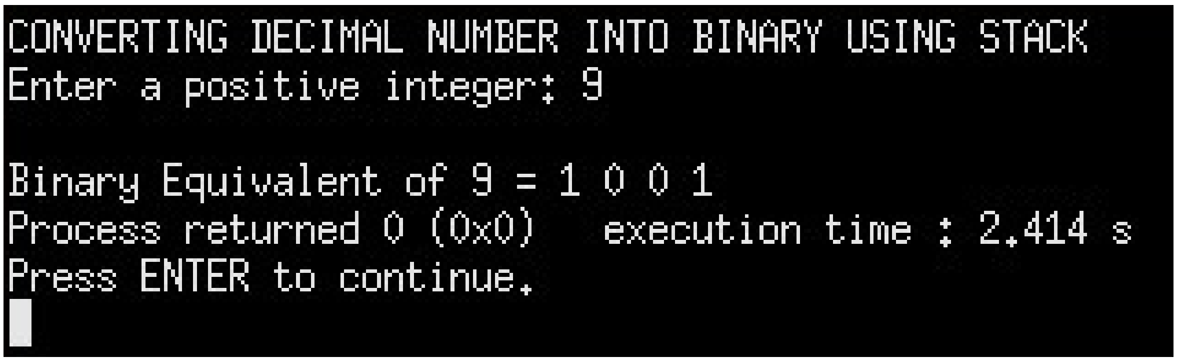
```
SELECT OPERATION ON THE STACK
1: PUSH
2: POP
3: DISPLAY
4: EXIT
Enter your choice: 2

STACK IS EMPTY :
1: PUSH
2: POP
3: DISPLAY
4: EXIT
Enter your choice: 1

Enter the element to push onto the stack: 2

Current content of the stack is: 2
1: PUSH
2: POP
3: DISPLAY
4: EXIT
Enter your choice: 1

Enter the element to push onto the stack: b

Current content of the stack is: 2 b
1: PUSH
2: POP
3: DISPLAY
4: EXIT
Enter your choice: 3

Current content of the stack is: 2 b
1: PUSH
2: POP
3: DISPLAY
4: EXIT
Enter your choice: 2

Popped item is = b
Current content of the stack is: 2
1: PUSH
2: POP
3: DISPLAY
4: EXIT
Enter your choice: █
```

2: Convert a given decimal number to binary using stack.

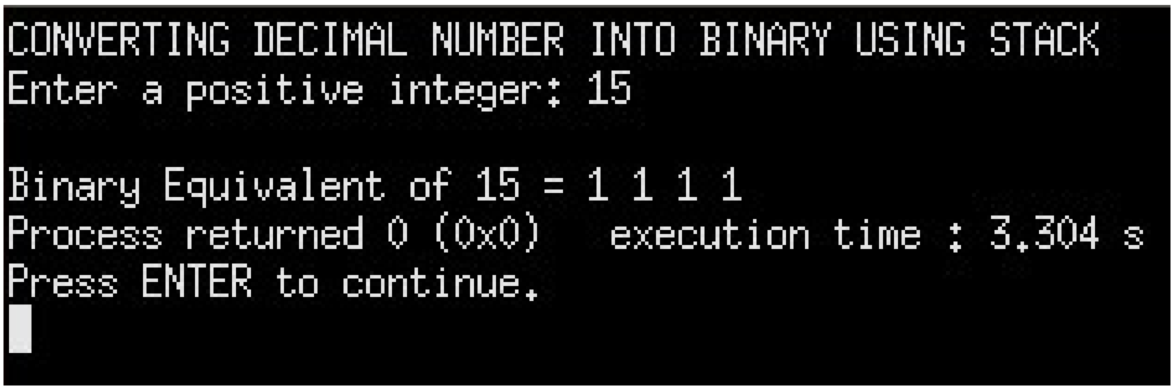
Code:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
int isEmpty(int top, int stack_arr[]);
void push(int x, int *top, int stack_arr[]);
int pop(int *top, int stack_arr[]);
void decimaltobinary(int number);
int main()
{
int number;
printf("CONVERTING DECIMAL NUMBER INTO BINARY USING STACK");
printf("\nEnter a positive integer: ");
scanf("%d",&number);
printf("\nBinary Equivalent of %d = ",number);
decimaltobinary(number);
return 0;}
void decimaltobinary(int number)
{
int stack[MAX], top=-1, rem;
while(number!=0)
{
rem = number%2;
push(rem, &top, stack);
number/=2;
}
while(top!=-1)
printf("%d ", pop(&top, stack));
}
void push(int x, int *top, int stack_arr[])
{
if(*top == (MAX-1))
printf("\nStack is full: ");
else
{
*top=*top+1;
stack_arr[*top] = x;
}
}
int pop(int *top, int stack_arr[])
{
int x;
if(*top == -1)
{
printf("\nStack is empty: ");
exit(1);
}
else
{
x = stack_arr[*top];
*top=*top-1;
}
return x;
}
```

Test Case:



```
CONVERTING DECIMAL NUMBER INTO BINARY USING STACK
Enter a positive integer: 9

Binary Equivalent of 9 = 1 0 0 1
Process returned 0 (0x0)   execution time : 2.414 s
Press ENTER to continue.
```



```
CONVERTING DECIMAL NUMBER INTO BINARY USING STACK
Enter a positive integer: 15

Binary Equivalent of 15 = 1 1 1 1
Process returned 0 (0x0)   execution time : 3.304 s
Press ENTER to continue.
```

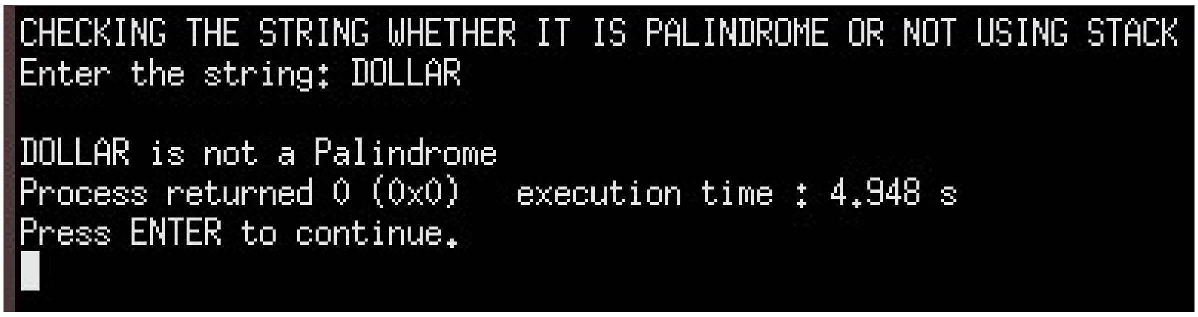
3: Determine whether a given string is palindrome or not using stack.

Code:

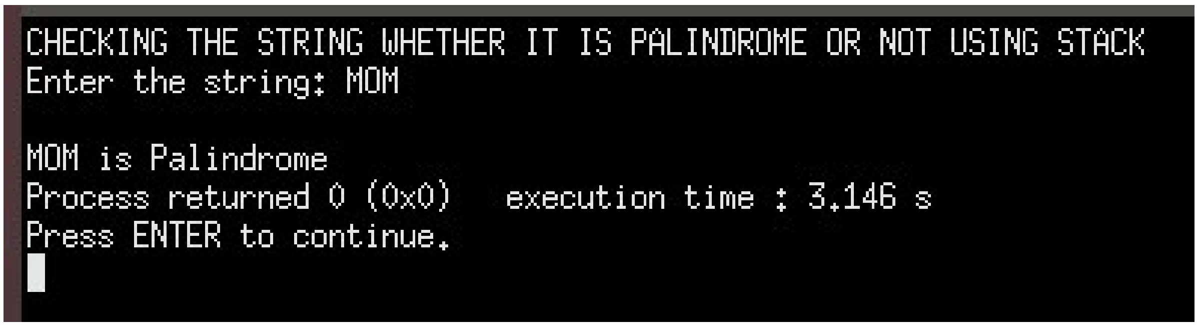
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 1000
#define EMPTY '\0'
typedef enum {
NO = 0,
YES = 1,
} BOOL;
BOOL isStackFull (int tos) {
if (tos == SIZE - 1)
return YES;
return NO;
}
```

```
BOOL isStackEmpty (int tos) {
if (tos == -1)
return YES;
return NO;
}
void push (char *stack, int item, int *tos) {
if (isStackFull (*tos)) {
printf("\nSTACK IS FULL");
return;
}
(*tos) += 1;
*(stack + (*tos)) = item;
}
char pop (char *stack, int *tos) {
if (isStackEmpty (*tos)) {printf("\nSTACK IS EMPTY");
return EMPTY;
}
return *(stack + ((*tos)--));
}
void display (char *stack, int tos) {
printf("\n");
char *pi;
for (pi = stack; pi <= stack + tos; ++pi)
printf("%c ", *pi);
}
BOOL isPalindrome (char *str) {
int tos = -1, i;
char *stack = (char *)malloc(sizeof(char *));
for (i = 0; i < strlen(str); ++i)
push(stack, str[i], &tos);
for (i = 0; i < strlen(str)/2; ++i)
if (pop(stack, &tos) != str[i])
return YES;
return NO;
}
int main(int argc, const char * argv[]) {
char *str = (char *)malloc(SIZE * sizeof(char));
printf("CHECKING THE STRING WHETHER IT IS PALINDROME OR NOT USING
STACK");
printf("\nEnter the string: ");
scanf("%s", str);
if (!isPalindrome (str))
printf("\n%s is Palindrome", str);
else
printf("\n%s is not a Palindrome", str);
return 0;
}
```

Test Case:



```
CHECKING THE STRING WHETHER IT IS PALINDROME OR NOT USING STACK
Enter the string: DOLLAR

DOLLAR is not a Palindrome
Process returned 0 (0x0)   execution time : 4.948 s
Press ENTER to continue.
```



```
CHECKING THE STRING WHETHER IT IS PALINDROME OR NOT USING STACK
Enter the string: MOM

MOM is Palindrome
Process returned 0 (0x0)   execution time : 3.146 s
Press ENTER to continue.
```