

Name: Aashutosh Kumar Pandit

Lab: 1

Roll No: CH.EN.U4CSE22076

Basic Lex Programs

1.Title: Write a program to check if a given number is prime or not.

Code:

```
%{  
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
%}  
  
%%  
[0-9]+ {  
    int num = atoi(yytext);  
  
    if(num <= 1) {  
        printf("%d is not prime.\n", num);  
    } else {  
        int i, flag = 1;  
        int limit = (int)sqrt(num);  
  
        for(i = 2; i <= limit; i++) {  
            if(num % i == 0) {  
                flag = 0;  
                break;  
            }  
        }  
        if(flag) {  
            printf("%d is prime.\n", num);  
        }  
    }  
}
```

```

        }
    }
    if(flag)
        printf("%d is prime.\n", num);
    else
        printf("%d is not prime.\n", num);
    }
}

```

`\n` ; // ignore new lines

`.` ; // ignore other characters

`%%`

```

int main() {
    printf("Enter a number: ");
    yylex();
    return 0;
}

```

```

int yywrap() {
    return 1;
}

```

Output:

```

asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ flex prime.l
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ gcc lex.yy.c -ll -lm -o prime_check
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./prime_check
Enter a number: 1
1 is not prime.

asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./prime_check
Enter a number: 2
2 is prime.

```

2.Title: Write a program to reverse a string without using built-in functions.

Code:

```
%{  
#include <stdio.h>  
  
void reverse(char *str, int length) {  
    int i;  
    for(i = 0; i < length / 2; i++) {  
        char temp = str[i];  
        str[i] = str[length - 1 - i];  
        str[length - 1 - i] = temp;  
    }  
}  
  
%}  
  
%%  
.*\n    {  
    // yytext contains the whole line including newline  
    int length = 0;  
    // Calculate length excluding newline  
    while(yytext[length] != '\n' && yytext[length] != '\0') {  
        length++;  
    }  
  
    // Reverse the string in yytext (modifying in place)
```

```

        reverse(yytext, length);

        // Add newline back manually
        yytext[length] = '\n';
        yytext[length+1] = '\0';

        printf("Reversed string: %s", yytext);
        return 0; // Stop after processing one line
    }
%%

int main() {
    printf("Enter a string: ");
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}

```

Output:

```

asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ flex reverse.l
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ gcc lex.yy.c -ll -o reverse_string
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./reverse_string
Enter a string: Aashutosh Kumar Pandit
Reversed string: tidnaP ramuK hsotuhSa

```

3.Title: Write a program to find the factorial of a number using recursion.

Code:

```
%{
#include <stdio.h>

// Recursive factorial function
long long factorial(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
}%

%%

[0-9]+ {
    int num = atoi(yytext);
    printf("Factorial of %d is %lld\n", num, factorial(num));
    return 0; // Stop after processing one number
}

\n    ; // ignore newline
.      ; // ignore any other characters

%%

int main() {
    printf("Enter a number: ");
    yylex();
```

```
        return 0;
    }

    int yywrap() {
        return 1;
    }
```

Output:

```
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ flex factorial.l
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ gcc lex.yy.c -ll -o factorial
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./factorial
Enter a number: 100
Factorial of 100 is 0
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./factorial
Enter a number: 10
Factorial of 10 is 3628800
```

4.Title: Write a program to find the largest and smallest element in an array.

Code:

```
%{
#include <stdio.h>
#include <limits.h>

int largest = INT_MIN;
int smallest = INT_MAX;
%}

%%

[0-9]+ {
    int num = atoi(yytext);
    if (num > largest)
        largest = num;
    if (num < smallest)
        smallest = num;
}

[\\n\\t ]+ ; // Ignore whitespace including newlines, tabs, spaces

. ; // Ignore any other characters
%%

int main() {
    printf("Enter numbers separated by space (Ctrl+D or Ctrl+Z to end
input):\\n");
```

```
yylex();  
printf("Largest element: %d\n", largest);  
printf("Smallest element: %d\n", smallest);  
return 0;  
}  
  
int yywrap() {  
    return 1;  
}
```

Output:

```
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ flex ls.l  
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ gcc lex.yy.c -ll -o ls  
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./ls  
Enter numbers separated by space (Ctrl+D or Ctrl+Z to end input):  
12 7 8 100 92 78 26  
Largest element: 100  
Smallest element: 7
```


5.Title: Write a program to find the sum of digits of a given number.

Code:

```
%{
#include <stdio.h>
%}

%%

[0-9]+ {
    int sum = 0;
    char *p = yytext;
    while (*p) {
        sum += (*p - '0'); // convert char digit to int and add
        p++;
    }
    printf("Sum of digits in %s is %d\n", yytext, sum);
    return 0; // stop after processing one number
}

\n    ; // ignore newlines
.      ; // ignore other characters
%%

int main() {
    printf("Enter a number: ");
    yylex();
    return 0;
}
```

```
int yywrap() {  
    return 1;  
}
```

Output:

```
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ flex sum.l  
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ gcc lex.yy.c -ll -o sum  
asecomputerlab@asecomputerlab-hp-prodesk-400-g7-micrtower-pc:~/Desktop/22076$ ./sum  
Enter a number: 234  
Sum of digits in 234 is 9
```