Week 3:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 100

struct stack {
    int top;
    unsigned capacity;
    char* array;
};

struct stack *createStack (unsigned capacity) {
    struct stack* stack = (struct stack *)
    malloc (sizeof (struct stack));
    stack -> top = -1;
    stack -> capacity = capacity;
    stack -> array = (char*) malloc (stack -> capacity
    of (char));
    return stack;
}

int isEmpty (struct stack* stack) {
    return stack -> top == -1;
}
```

```
void push(struct stack * stack, char op) {
    stack->array[++stack->top] = op;
}

char pop(struct Stack *stack) {
    if (!isEmpty(stack)) {
        return stack->array[stack->top--];
    }
    return '\0';
}

char peek(struct Stack* stack){
    if (!isEmpty(stack)) {
        return stack->array[stack->top];
    }
    return '\0';
}

int isOperator(char ch){
    return (ch=='+'||ch=='-'||ch=='*'||
    ch=='/');
}

int presedence(char op){
    switch(op){
        case '+';
        case '-';
            return 1;
```

```
        case '*':
        case '/':
            return 2;
        default:
            return -1;
    }
}

void infixToPostfix(char* infix, char* postfix)
{
    struct Stack* stack = createStack(MAX_SIZE);
    int i, j;

    for(i=0, j=0; infix[i]; ++i){
        if(infix[i] == '(' || infix[i] == '\t')
        {
            continue;
        }
        else if(isdigit(infix[i]) || isalpha(infix[i])){
            postfix[j++] = infix[i];
        }
        else if(isOperator(infix[i])){
            while(!isEmpty(stack) && precedence(infix[i])
            <= precendence(peek(stack))){
                postfix[j++] = pop(stack);
            }
```

```
        push(stack, infix[i]);
    }
    else if (infix[i] == '('){
        push(stack, infix[i]);
    }
    else if (infix[i] == ')'){
        while (!isEmpty(stack) && peek(stack) !=
        '('){
            postfix[j++] = pop(stack);
        }
        pop(stack);
    }

}
while(!isEmpty(stack)){
    postfix[j++] = pop(stack);
}
postfix[j] = '\0';
}

int main() {
    char infix[MAX_SIZE], postfix[MAX_SIZE];
    printf("Enter infix expression ");
    fgets(infix, sizeof(infix), stdin);
    infixToPostfix(infix, postfix);
    printf("Postfix expression : %s\n", postfix);
    return 0;
}
```

Output

Enter infix expression = a+b+c/d^e

Postfix expression : ab+cd/e^+

18/1/24