Utils

- Stack
- Insertation Sort

```
class Stack {
    public:
        int top;
        int size;
        int *arr;

    Stack(int size) {
        this->top = -1;
        this->size = size;
        this->arr = new int[size];
    }

    bool isEmpty() {
        if (this->top == -1) {
            return true;
        }

        return false;
    }

    bool isFull() {
        if (this->top == this->size - 1) {
            return true;
        }

        return false;
    }

    void push(int val) {
        if (isFull()) {
            printf("StackOverflowed");
        } else {
            this->arr[++top] = val;
        }
    }

    int pop() {
        if (isEmpty()) {
            cout << "StackUnderflow" << endl;
            return -1;
        }
        return this->arr[top--];
    }
    int getSize() {
        return this->size;
    }

    int peek (int pos) {
        if (isEmpty()) {
            return -1;
        }
        return this->arr[this->top - pos + 1];
    }

    int getTop() {
        return this->top;
    }
};
```

```
void insertion_sort(int *a, int len)
{
    for (int i = 1; i < len; i++)
    {
        int current = a[i];
        int j = i - 1;
        while (a[j] > current && j >= 0)
        {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = current;
    }
}
```

Q2. Write a program from scratch that reverses the words in a sentence.

```
int main() {
   char str[] = "Hello World CPP";

   char *word = strtok(str, " ");

   while (word != NULL) {
       for (int i = strlen(word); i > 0; --i) {
           cout << word[i - 1];
       }
       cout << " ";
       word = strtok(NULL, " ");
   }
}
```

Q4. Write the function transforming a decimal number into a binary number by using stack

```
int main() {
    int r;
    Stack s = Stack(5);
    int num = 100;

    while (num > 0) {
        r = num % 2;
        s.push(r);
        num = num / 2;
    }

    while (!(s.isEmpty())) {
        cout << s.pop() << endl;
    }
}
```

Q5. Write the function that removes all even numbers from the given stack. The mutual order of odd numbers must stay unchanged. The function returns the number of removed numbers.

```
int main() {
    Stack stack = Stack(10);
    Stack temp = Stack(10);
    int removeCounter = 0, ele;

    stack.push(7);
    stack.push(8);
    stack.push(9);
    stack.push(10);

    while (!stack.isEmpty()) {
        ele = stack.pop();
        if (ele % 2 != 0) {
            temp.push(ele);
        } else {
            removeCounter++;
        }
    }

    while (!temp.isEmpty()) {
        stack.push(temp.pop());
    }

    cout << "Total Even Integers Removed: " << removeCounter << endl;

    cout << "After Removing Even Integers" << endl;
    while (!stack.isEmpty()) {
        cout << stack.pop() << " ";
    }
}
```

Q6. Write the function that returns duplicate stack of the given stack. Duplicate stack contains the same elements as the original stack, and in the same order. The original stack must stay unchanged.

```
#include "iostream"
#include "Stack.cpp"

Stack duplicateStack(Stack stack) {
    int size = stack.getSize();
    Stack dulStack = Stack(size);

    for (int i = stack.getTop() + 1; i > 0; i--) {
        dulStack.push(stack.peek(i));
    }
    return dulStack;
}

int main() {
    Stack stack = Stack(10);

    stack.push(1);
    stack.push(2);
    stack.push(3);
    stack.push(4);
    stack.push(5);

    Stack dulStack = duplicateStack(stack);

    cout << "Original Stack" << endl;
    cout << "Size: " << stack.getSize() << endl;
    cout << "Top: " << stack.getTop() << endl;
    cout << "Elements: ";

    while (!stack.isEmpty()) {
        cout << stack.pop() << " ";
    }

    cout << "\n\n";

    cout << "Duplicate Stack" << endl;
    cout << "Size: " << dulStack.getSize() << endl;
    cout << "Top: " << dulStack.getTop() << endl;
    cout << "Elements: ";

    while (!dulStack.isEmpty()) {
        cout << dulStack.pop() << " ";
    }
}
```

Q8. A palindrome is a phrase that reads the same forward and backward (examples: ‘racecar’, ‘radar’, ‘noon’, or ‘rats live on no evil star’). By extension we call every string a palindrome that reads the same from left to right and from right to left. Develop a recursive algorithm that takes as input a string and decides whether the string is a palindrome. Write down your algorithm in pseudocode.

```
bool findpl(string plStr, int i) {
    if (i > plStr.size() / 2) {
        return true;
    }
    return plStr[i] == plStr[plStr.size() - i - 1] && findpl(plStr, i + 1);
}

int main() {
    char plStr[] = "noon";
    if(findpl(plStr, 0)) {
        cout << "Yes";
    } else {
        cout << "No";
    }
}
```

Q9. Write a program to remove the duplicate elements of a given array and return the new length of the array. Sample array: [20, 20, 30, 40, 50, 50, 50] After removing the duplicate elements, the program should return 4 as the new length of the array.

```
int main() {
    int arr[] = { 20, 20, 30, 40, 50, 50, 50 };
    int n = sizeof arr / sizeof(int);
    int count = 0;

    insertion_sort(arr, n);

    for (int i = 0; i < n - 1; i++) {
        if (arr[i] != arr[i + 1]) {
            count++;
        }
    }

    count++;
    cout << count ;
}
```

Q10. Write a program for Matrix multiplication of two matrices having different sizes?

```
int main() {
    int mat1[3][4] = {{1, 2, 3, 6},
                      {4, 5, 6, 4},
                      {4, 5, 6, 4}};

    int mat2[3][3] = {{1, 2, 3},
                      {4, 5, 6},
                      {6, 7, 8}};


    int r1 = sizeof(mat1) / sizeof (mat1[0]);
    int r2 = sizeof(mat2) / sizeof (mat2[0]);
    int c2 = sizeof(mat2[0]) / sizeof(mat2[0][0]);

    if (r1 != c2) {
        cout << "Impossible, Even For Super Computers" << endl;
        exit(1);
    }

    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            int sum = 0;
            for (int k = 0; k < r2; k++) {
                sum = sum + mat1[i][k] * mat2[k][j];
            }
            cout << sum << " ";
        }
        cout << "\n";
    }
}
```