Пояснение:

- Реализирайте задачите спазвайки добрите ООП практики (валидация на данните, подходяща капсулация и тн.)
- Решение, в които не са спазени ООП принципите ще бъдат оценени с 0 точки.
- Предадените от вас решения трябва да могат да се компилират успешно на Visual C++.
- **Не е разрешено** да ползвате библиотеки от STL и STL функции.

Изисквания за предаване:

- Всички задачи ще бъдат проверени автоматично за преписване. Файловете с голямо съвпадение ще бъдат проверени ръчно и при установено плагиатство ще бъдат анулирани.
- Предаване на домашното в указания срок от всеки студент като .zip архив със следното име:

  (номер_на_домашно)_SI_(курс)_(група)_(факултетен_номер)

  - (номер_на_домашно) е цяло число, отговарящо на номера на домашното за което се отнася решението (например 1);
  - (курс) е цяло число, отговарящо на курс (например 1);
  - (група) е цяло число, отговарящо на групата Ви (например 1);
  - (факултетен_номер) е цяло число, отговарящо на факултетния Ви номер (например 12345);

Пример за .zip архив за домашно: 2_SI_1_1_12345.zip

Архивът да съдържа само изходен код (.cpp и .h/.hpp файлове) с решение отговарящо на условията на задачите, като файловете изходен код за всяка задача трябва да са разположени в папка с име (номер_на_задача).

Качване на архива на посоченото място в Moodle

Задача 1 - Small string optimization

На лекции говорихме за **SSO** - за стрингове с до определена дължина не е необходимо да се заделя динамична памет (байтовете за символите се пазят в указателя и променливата за дължината). Модифицирайте написания в час стринг или имплементирайте ваш стринг, в който при достатъчно малки стрингове се прилага SSO.
Забележка 1: При имплементиране на ваш стринг, можете да добавите (ако прецените) променлива за капацитет (capacity) за по-бързи конкатенации, но в обекта не трябва да се заема допълнително памет освен за указателя(char*), размера и (евентуално)капацитета.
Забележка 2: Опишете в отделен файл кратка документация при какви стрингове се прилага оптимизацията и как запазвате информацията в променливите.

Задача 2 - StringCreaterPieceByPiece

Имплементирайте клас за построяване на стринг без да алокираме памет за всеки добавен символ. Вашият клас трябва да работи с “парчета” (StringPiece) - част от стринга с дължина най-много 16 символа.
Във всяко едно от тези парчета трябва да имате функции/оператори за:
- get и set за стойността.
- Оператори << и >> за конкатенация на низове/числа в началото/края.
- Премахване на първите/последните k символа.
- Промяна на символ по индекс.
Във вашия клас трябва да имате колекция от такива парчета като на всяко от тях трябва да можете да им прилагате по-горе описаните операции. В конструктора се подава начален капацитет за броя парчета.
Трябва да имате и операции за:
- Добавяне на ново парче в края на колекцията. Ако не се подаде низ, то парчето да е със стойност празния стринг.
- Размяна на две парчета по подадени два индекса.
- Изтриване на парче по индекс. Този индекс остава празен.
- Добавяне на парче по индекс. Ако на този индекс има парче, то да се замести с новото парче.
- Взимане дължината на построения стринг до момента.
- Връщане на обект от тип MyString (или ваш написан стринг) с точна големина с построения стринг до момента. Построеният стринг е конкатенацията на стойностите на всички парчета. Ако индекс (ПРЕДИ последното парче) е празен (парчето е било е изтрито и не е добавени нищо), то в стринга се поставят 20 символа за интервал.

```
StringCreaterPieceByPiece sc(2);

sc.addPiece(“test”);
sc.addPiece();
sc.addPiece(“football”);

sc[1] << “ friends ”;
“ Hello ” >> sc[1];
10 >> sc[2];

MyString result1 = sc.getString(); // “test Hello friends 10football”

sc.swap(1,2);
MyString result2 = sc.getString(); // “test10football Hello friends”

sc.remove(1);
MyString result3 = sc.getString(); // “test10                    Hello friends”
```