*W wyniku wykonania poniższych zadań powinien powstać **program**, którego kody źródłowe powinny zostać przesłane na adres antyplagiatu (informacja w ramce na końcu zadania).*

Naszym zadaniem będzie napisanie prostego menadżera pamięci. Menadżer powinien zostać napisany w formie biblioteki statycznej i będzie udostępniał dwie funkcje:

- void ***mem_allock**(void ***ptr**, unsigned int **size**)

- int **mem_free**(void ***ptr**)

Do implementacji menadżera wykorzystujemy funkcje biblioteczne: malloc, calloc, realloc, free. Menadżer przez cały czas życia procesu powinien utrzymywać informacje o wszystkich zaalokowanych za pomocą funkcji mem_alloc fragmentach pamięci (np. w formie listy) i w razie konieczności zwolnić je automatycznie (atexit).

Opis działania funkcji:

1. **mem_alloc** przyjmuje wskaźnik na miejsce w pamięci **ptr** i rozmiar alokowanego obszaru w bajtach **size**. Jeżeli **ptr** jest ustawione na **NULL**, to alokowany jest nowy obszar rozmiaru **size**. W przypadku gdy **ptr** zawiera adres (inny niż **NULL**) ma zostać sprawdzone, czy adres jest prawidłowy (tzn. czy menadżer ma go na liście swoich alokacji) realokować ten obszar do nowego rozmiaru **size**. Funkcja zwraca adres początku zaalokowanego lub realokowanego obszaru w przypadku powodzenia, **NULL** w przypadku błędu.
   Uwaga! Nowo alokowana pamięć powinna być od razu czyszczona (wszystkie bajty ustawiane na 0).
2. **mem_free** przyjmuje wskaźnik na zaalokowany przez menadżera obszar pamięci. Jeżeli menadżera rozpozna adres jako poprawny to zwalnia obszar i zwraca 0, w przeciwnym wypadku zwraca -1.

W obydwu funkcjach uwzględnić możliwość wystąpienia błędów funkcji malloc, calloc, realloc i free zwracając unikalne kody błędów. Rozważyć użycie na poziomie bibliotek zmiennej extern, która zawierałaby rozszerzony kod błędu.

Przygotować program korzystający z menadżera, który zademonstruje różne scenariusze działania menadżera, przeanalizować jego skuteczność z użyciem programu Valgrind.

Przed wysłaniem plików źródłowego na antyplagiat ich nazwy zmieniamy na:
numer_indeksu.ps.lab06.libmem.c,
numer_indeksu.ps.lab06.libmem.h,
numer_indeksu.ps.lab06.main.c,
(czyli np. 66666.ps.lab06.main.c).

Kody źródłowe (3 plik) po oddaniu prowadzącemu zajęcia laboratoryjne muszą zostać jako załączniki przesłane na adres **pss1@zut.edu.pl** (wysyłamy jeden mail z trzema załącznikami):

- pliki z kodami źródłowymi muszą mieć nazwę zgodne ze wzorcem podanym w treści zadania,

- mail musi zostać wysłany z poczty uczelnianej (domena **zut.edu.pl**),
- temat maila musi mieć postać: **PS IS1 999X LAB06**, gdzie 999X to numer grupy laboratoryjnej (np. PS IS1 321 LAB06),
- w pierwszych trzech liniach kodu źródłowego w komentarzach (każda linia komentowana osobno) musi znaleźć się:
    - informacja identyczna z zamieszczoną w temacie maila,
    - imię i nazwisko osoby wysyłającej maila,
    - adres e-mail, z którego wysłano wiadomość,
      np.:

      ```
      // PS IS1 321 LAB06
      // Jan Nowak
      // nj66666@zut.edu.pl
      ```

- e-mail nie może zawierać żadnej treści (tylko załączniki).

Dostarczone kody programów będą analizowane pod kątem wykrywania plagiatów. Niewysłanie wiadomości, wysłanie jej w formie niezgodnej z powyższymi wymaganiami lub wysłanie pliku, który nie będzie się kompilował i uruchamiał, będzie traktowane jako brak programu i skutkowało otrzymaniem za niego oceny niedostatecznej.