Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Мультипарадигмене програмування

ЗВІТ

до лабораторних робіт

Виконав

студент ІП-01 Храмченко А.С. _____

(№ групи, прізвище, ім’я, по батькові) _____

Прийняв ас. Очеретяний О. К. _____

(посада, прізвище, ім’я, по батькові) _____

Київ 2022

1. Завдання лабораторної роботи

Ви напишете 11 функцій SML (і тести для них), пов’язаних з календарними датами. У всіх завданнях, “дата” є значенням SML типу int*int*int, де перша частина - це рік, друга частина - місяць і третя частина - день. «Правильна» дата має позитивний рік, місяць від 1 до 12 і день не більше 31 (або 28, 30 - залежно від місяця). Перевіряти “правильність” дати не обов'язково, адже це досить складна задача, тож будьте готові до того, що багато ваших функцій будуть працювати корректно для деяких/всіх “неправильних” дат у тому числі. Також, «День року» — це число від 1 до 365 де, наприклад, 33 означає 2 лютого. (Ми ігноруємо високосні роки, за винятком однієї задачі.)

1. Напишіть функцію is_older, яка приймає дві дати та повертає значення true або false. Оцінюється як true, якщо перший аргумент - це дата, яка раніша за другий аргумент. (Якщо дві дати однакові, результат хибний.)
2. Напишіть функцію number_in_month, яка приймає список дат і місяць (тобто int) і повертає скільки дат у списку в даному місяці.
3. Напишіть функцію number_in_months, яка приймає список дат і список місяців (тобто список int) і повертає кількість дат у списку дат, які знаходяться в будь-якому з місяців у списку місяців. Припустимо, що в списку місяців немає повторюваних номерів. Підказка: скористайтеся відповіддю до попередньої задачі.
4. Напишіть функцію dates_in_month, яка приймає список дат і число місяця (тобто int) і повертає список, що містить дати з аргументу “список дат”, які знаходяться в переданому місяці. Повернутий список повинен містять дати в тому порядку, в якому вони були надані спочатку.
5. Напишіть функцію dates_in_months, яка приймає список дат і список місяців (тобто список int) і повертає список, що містить дати зі

списку аргументів дат, які знаходяться в будь-якому з місяців у списку місяців. Для простоти, припустимо, що в списку місяців немає повторюваних номерів. Підказка: Використовуйте свою відповідь на попередню задачу та оператор додавання списку SML (@).

6. Напишіть функцію get_nth, яка приймає список рядків і int n та повертає n-й елемент списку, де голова списку є першим значенням. Не турбуйтеся якщо в списку занадто мало елементів: у цьому випадку ваша функція може навіть застосувати hd або tl до порожнього списку, і це нормально.
7. Напишіть функцію date_to_string, яка приймає дату і повертає рядок у вигляді “February 28, 2022” Використовуйте оператор ^ для конкатенації рядків і бібліотечну функцію Int.toString для перетворення int в рядок. Для створення частини з місяцем не використовуйте купу розгалужень. Замість цього використайте список із 12 рядків і свою відповідь на попередню задачу. Для консистенції пишіть кому після дня та використовуйте назви місяців англійською мовою з великої літери.
8. Напишіть функцію number_before_reaching_sum, яка приймає додатний int під назвою sum, та список int, усі числа якої також додатні. Функція повертає int. Ви повинні повернути значення int n таке, щоб перші n елементів списку в сумі будуть менші sum, але сума значень від n + 1 елемента списку до кінця був більше або рівний sum.
9. Напишіть функцію what_month, яка приймає день року (тобто int між 1 і 365) і повертає в якому місяці цей день (1 для січня, 2 для лютого тощо). Використовуйте список, що містить 12 цілих чисел і вашу відповідь на попередню задачу.
10. Напишіть функцію month_range, яка приймає два дні року day1 і day2 і повертає список int [m1,m2,...,mn] де m1 – місяць day1, m2 –

місяць day1+1, ..., а mn – місяць day2. Зверніть увагу, що результат матиме довжину day2 - day1 + 1 або довжину 0, якщо day1>day2.

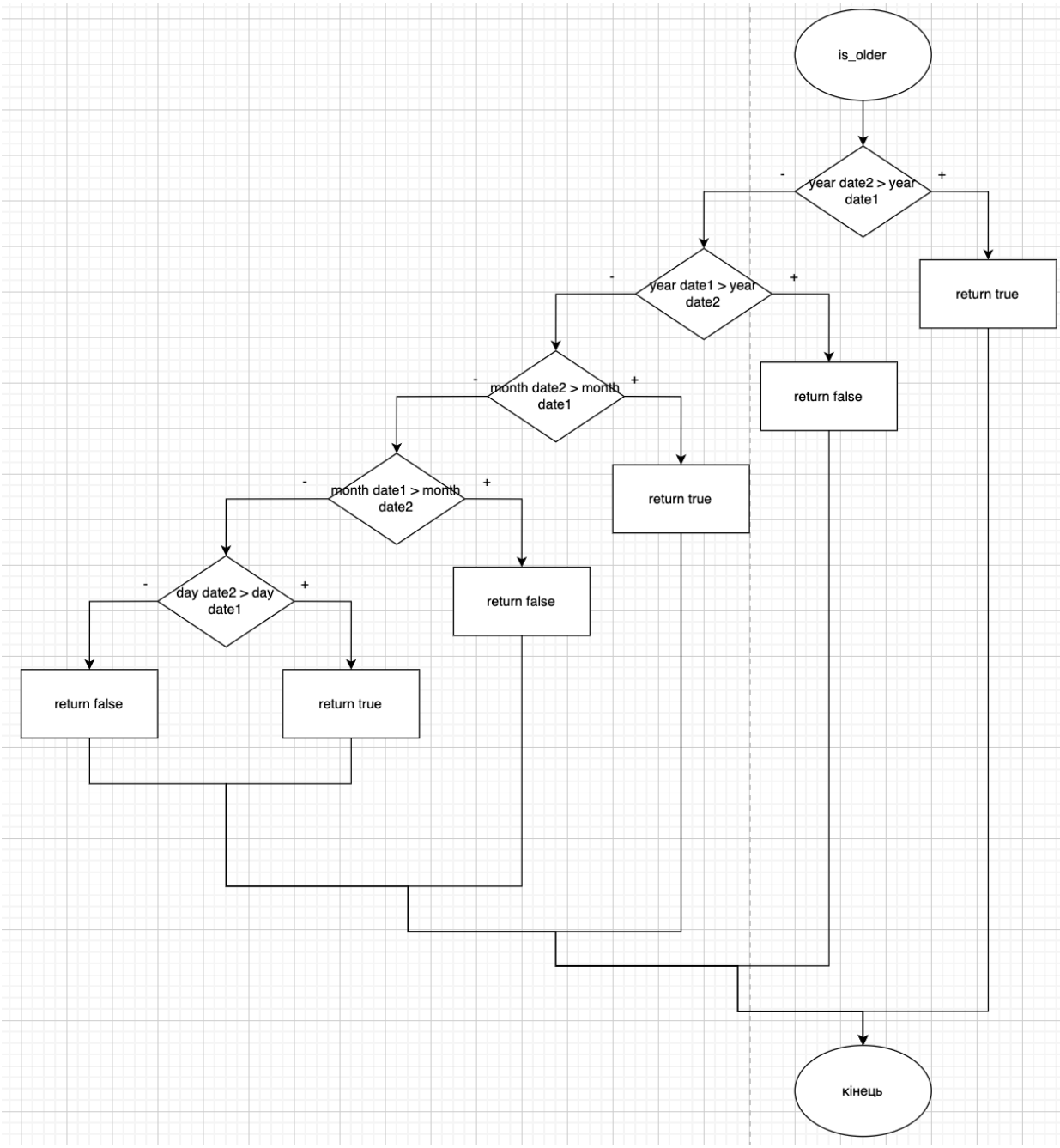
11. Напишіть найстарішу функцію, яка бере список дат і оцінює параметр (int*int*int). Він має оцінюватися як NONE, якщо список не містить дат, і SOME d, якщо дата d є найстарішою датою у списку.

2. Опис використаних технологій

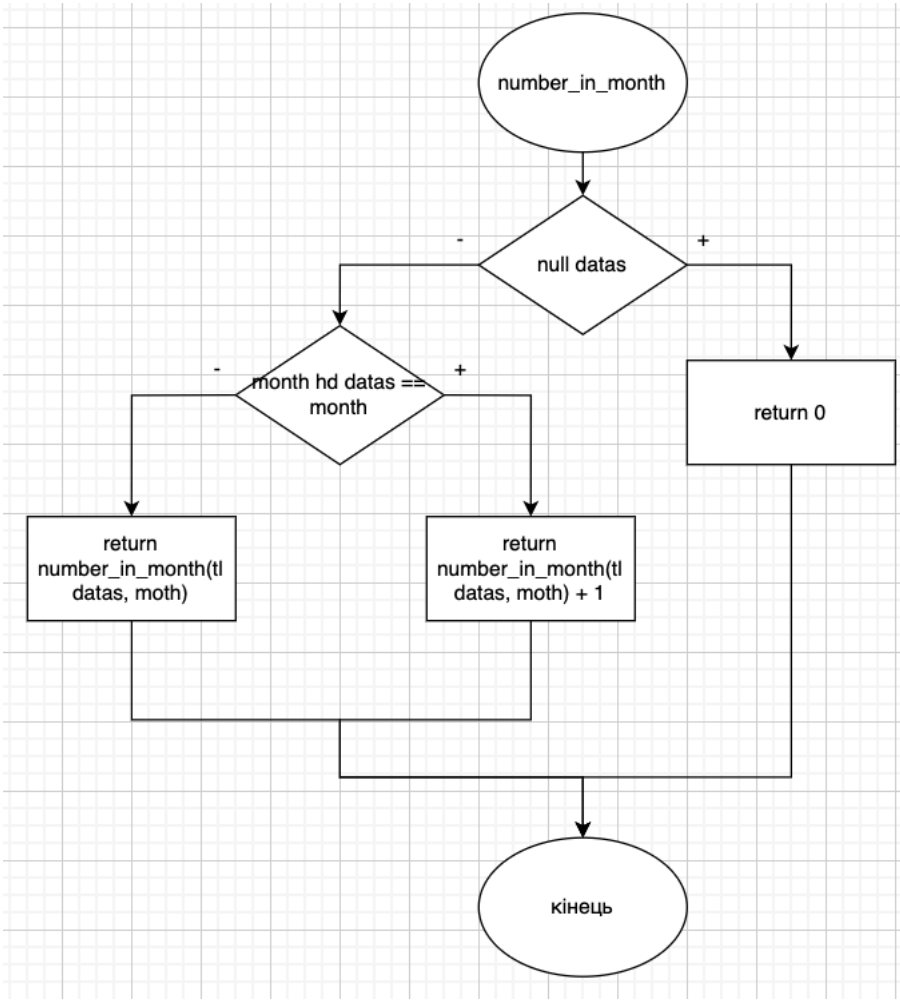
Для виконання цього завдання було використано мову програмування SML, як і зазначено у завданні

3. Опис алгоритму

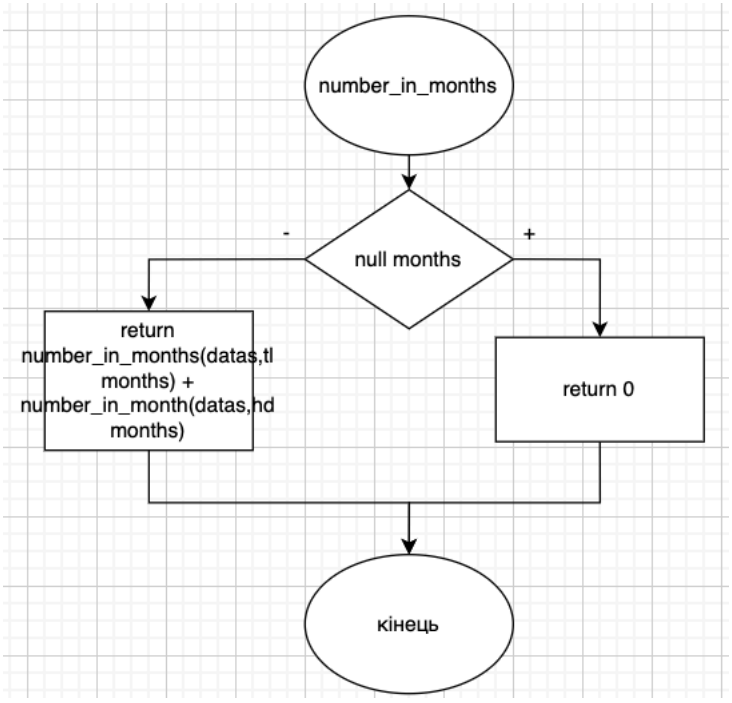
Завдання 1


is_older
year date2 > year date1
-
+
return true
year date1 > year date2
-
+
return false
month date2 > month date1
-
+
return true
month date1 > month date2
-
+
return false
day date2 > day date1
-
+
return false
return true
кінець

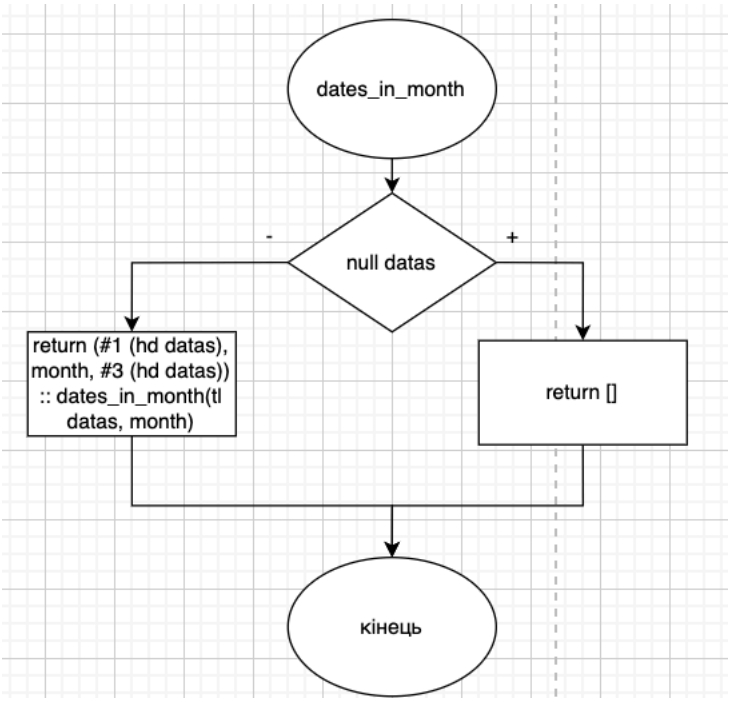
Завдання 2



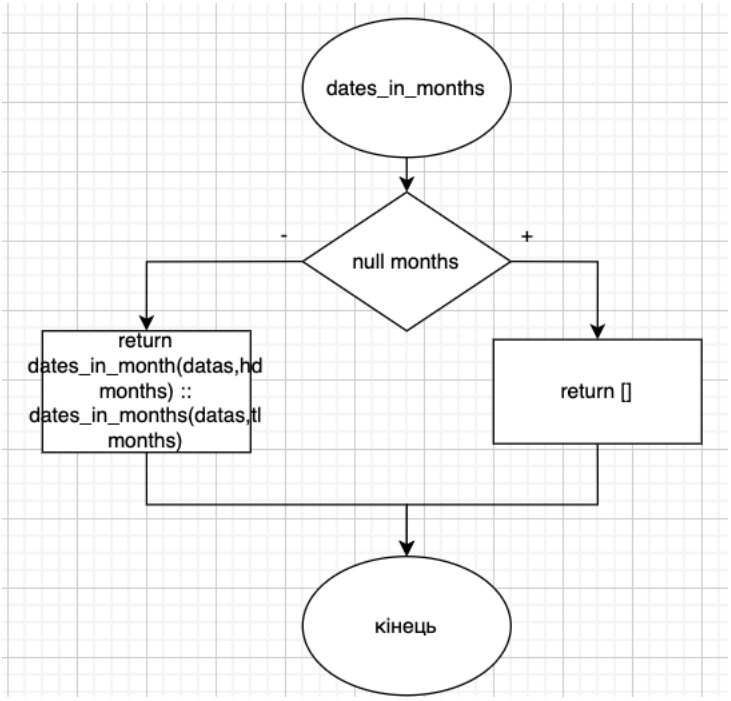
Завдання 3



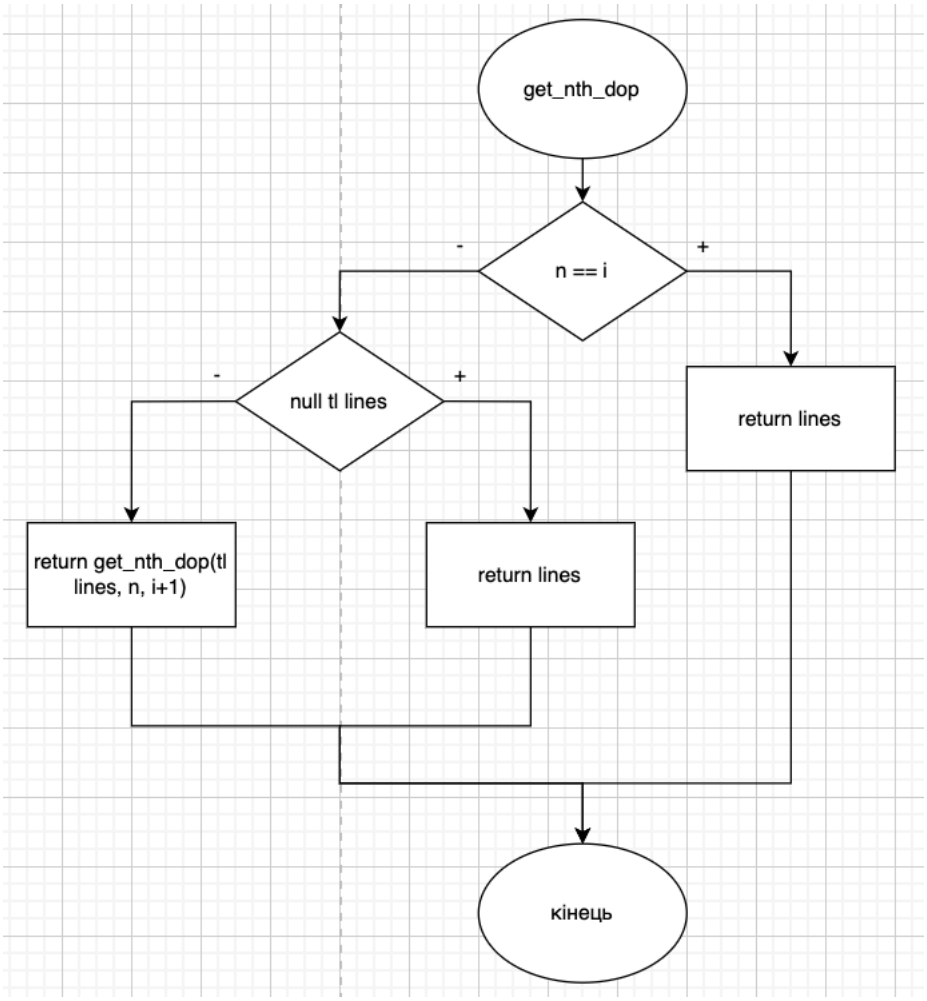
Завдання 4

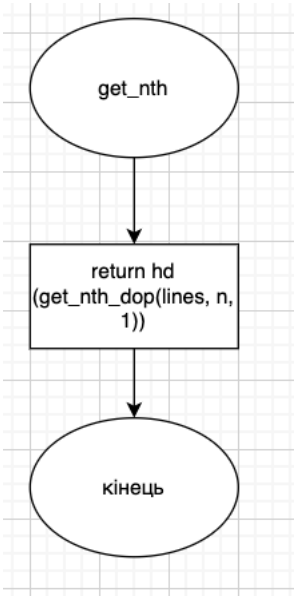


Завдання 5

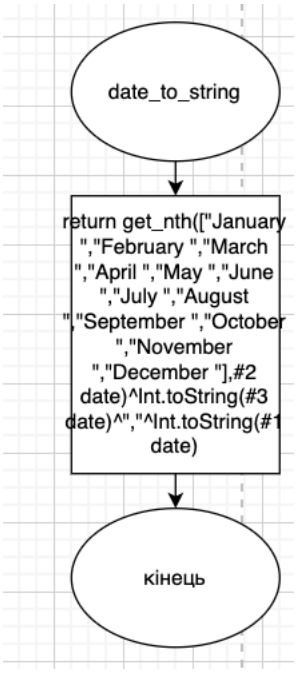


Завдання 6

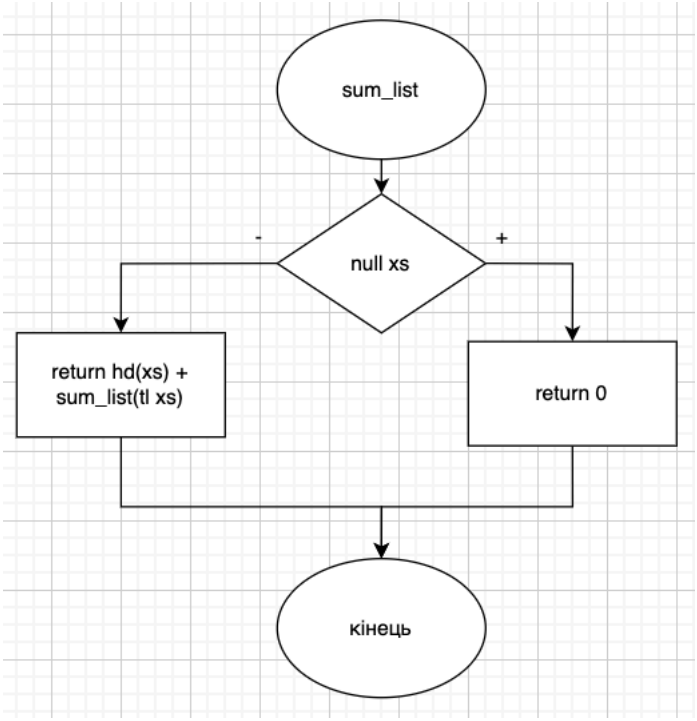

get_nth_dop
n == i
-
+
null tl lines
-
+
return lines
return get_nth_dop(tl lines, n, i+1)
return lines
кінець

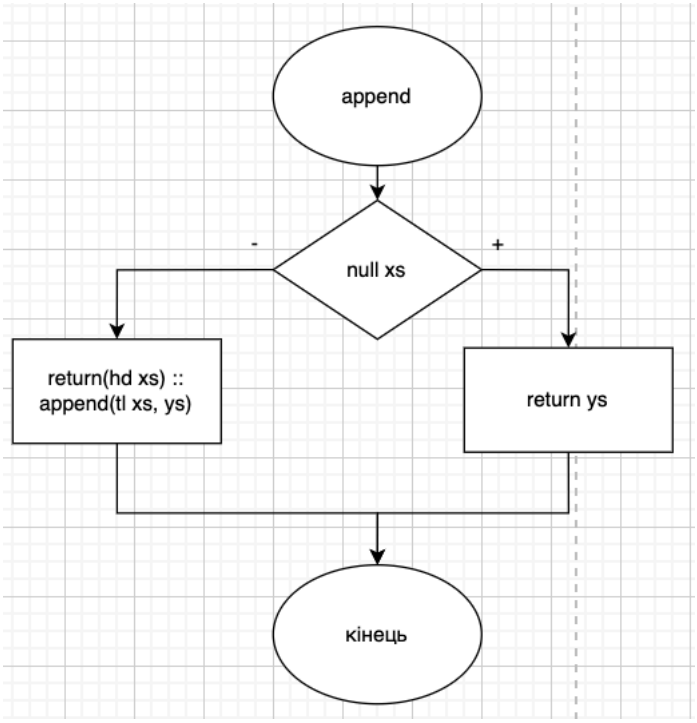

get_nth
return hd (get_nth_dop(lines, n, 1))
кінець

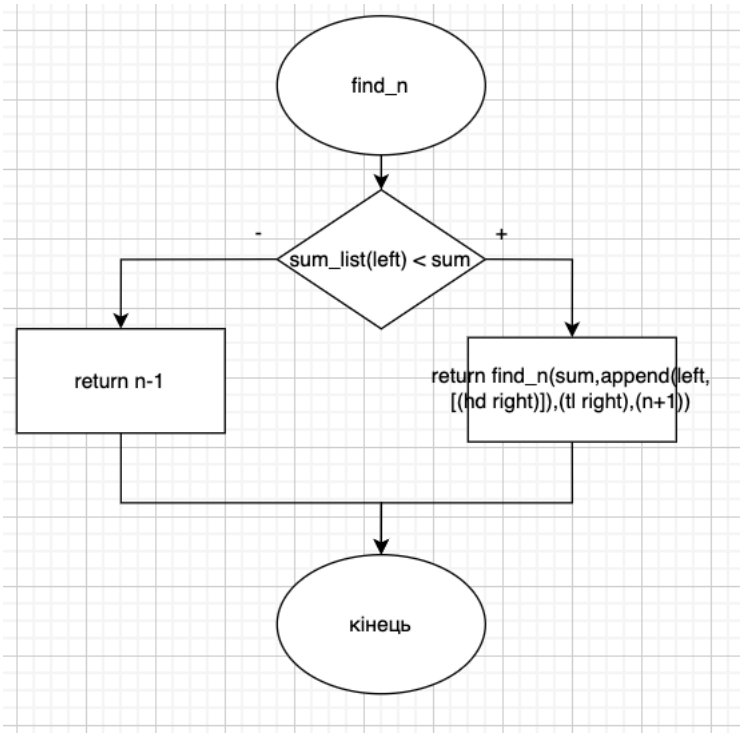
Завдання 7

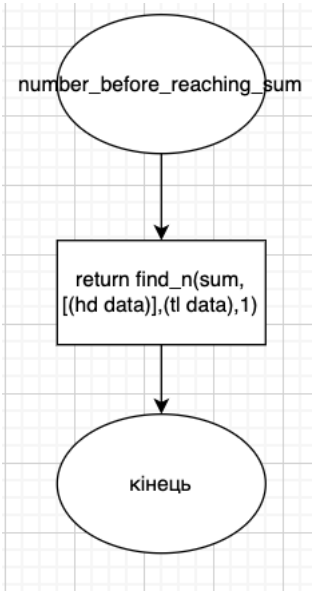

date_to_string
return get_nth(["January ","February ","March ","April ","May ","June ","July ","August ","September ","October ","November ","December "],#2 date)^Int.toString(#3 date)^","^Int.toString(#1 date)
кінець

Завдання 8

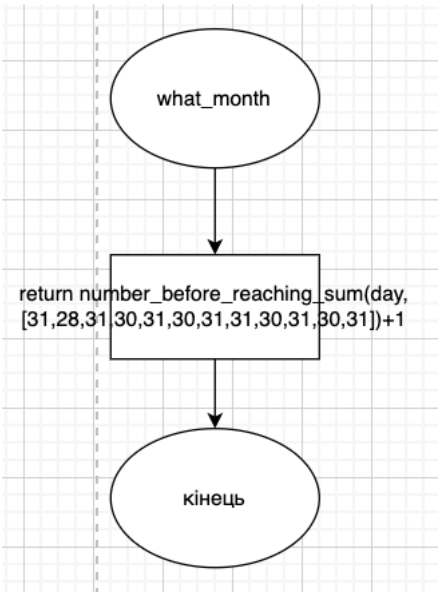

sum_list
null xs
-
+
return hd(xs) + sum_list(tl xs)
return 0
кінець


append
null xs
-
+
return(hd xs) ::
append(tl xs, ys)
return ys
кінець

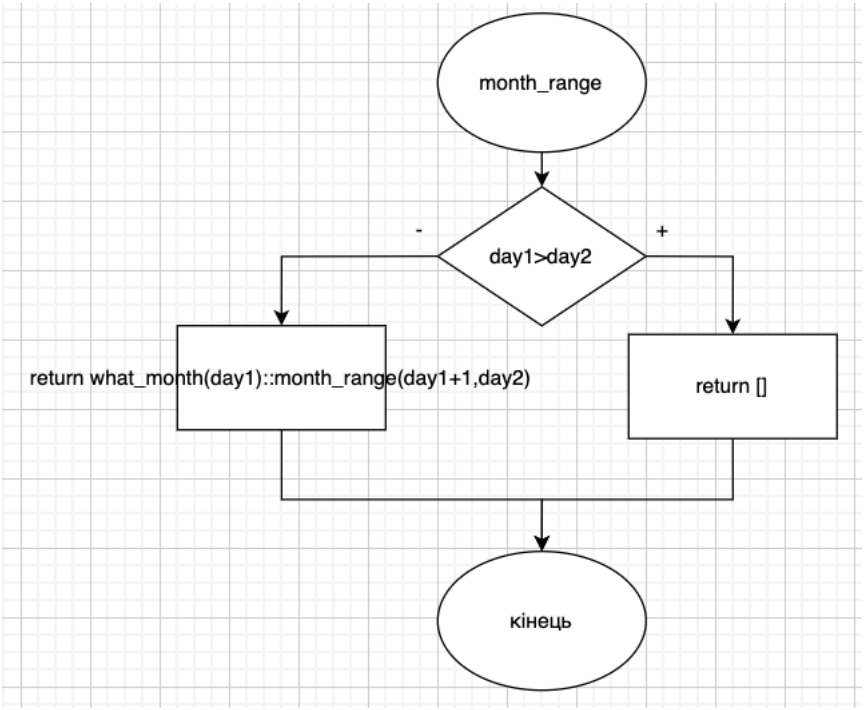

find_n
sum_list(left) < sum
-
+
return n-1
return find_n(sum,append(left,
[(hd right)]),(tl right),(n+1))
кінець



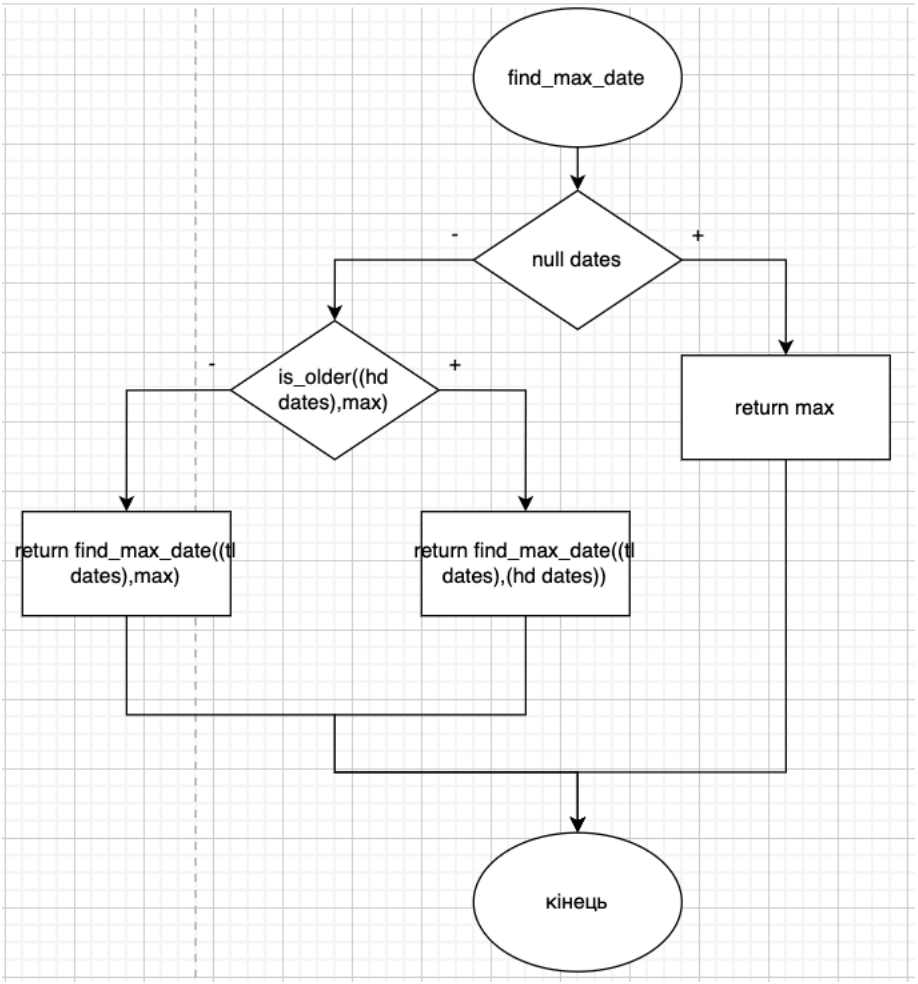
Завдання 9

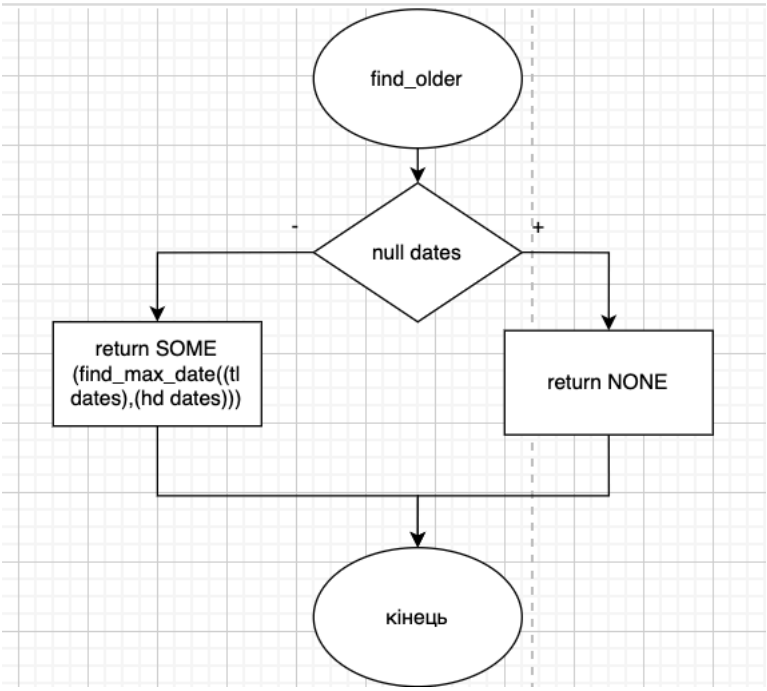


Завдання 10


month_range
day1>day2
-
+
return what_month(day1)::month_range(day1+1,day2)
return []
кінець

Завдання 11


find_max_date
null dates
-
+
is_older((hd dates),max)
-
+
return max
return find_max_date((tl dates),max)
return find_max_date((tl dates),(hd dates))
кінець



4. Опис програмного коду

```
fun is_older (date1:int*int*int, date2:int*int*int) =
  if (#1 date2) > (#1 date1)
    then true
    else if (#1 date1) > (#1 date2)
      then false
      else if (#2 date2) > (#2 date1)
        then true
        else if (#2 date1) > (#2 date2)
          then false
          else if (#3 date2) > (#3 date1)
            then true
            else false;

fun number_in_month (datas:(int*int*int) list, month:int) =
  if null datas
    then 0
    else if #2 (hd datas) = month
      then number_in_month(tl datas, month) + 1
      else number_in_month(tl datas, month);

fun number_in_months (datas:(int*int*int) list, months:int list) =
  if null months
    then 0
    else number_in_months(datas, tl months)+number_in_month(datas, hd months);

fun dates_in_month (datas:(int*int*int) list, month:int) =
  if null datas
    then []
    else (#1 (hd datas), month, #3 (hd datas)) :: dates_in_month(tl datas, month);

fun dates_in_months (datas:(int*int*int) list, months:int list) =
  if null months
    then []
    else dates_in_month(datas,hd months) :: dates_in_months(datas,tl months);
```

```
fun get_nth_dop (lines:string list, n:int, i:int) =
  if n=i
    then lines
    else if null (tl lines)
      then lines
      else get_nth_dop(tl lines, n, i+1);

fun get_nth (lines:string list, n:int) = (hd (get_nth_dop(lines, n, 1)));

fun date_to_string (date:int*int*int) = get_nth(["January ","February ","March ","April ","May ","June "

fun sum_list (xs : int list) =
  if null xs
    then 0
    else hd(xs) + sum_list(tl xs);

fun append (xs : int list, ys : int list) =
   if null xs
    then ys
    else (hd xs) :: append(tl xs, ys);

fun find_n (sum:int, left: int list, right: int list, n:int) =
  if (sum_list(left) < sum)
    then find_n(sum,append(left,[(hd right)]),(tl right),(n+1))
    else n-1;

fun number_before_reaching_sum (sum:int, data:int list) = find_n(sum,[(hd data)],(tl data),1);

fun what_month(day:int) = number_before_reaching_sum(day,[31,28,31,30,31,30,31,31,30,31,30,31])+1;

fun month_range(day1:int, day2:int) =
  if day1>day2
    then []
    else what_month(day1)::month_range(day1+1,day2);
```

```
fun find_max_date(dates:(int*int*int) list,max:int*int*int) =
  if null dates
    then max
    else if is_older((hd dates),max)
      then find_max_date((tl dates),(hd dates))
      else find_max_date((tl dates),max);

fun find_older(dates:(int*int*int) list) =
  if null dates
    then NONE
    else SOME (find_max_date((tl dates),(hd dates)));
```

5. Тестування програми

Тести

```
is_older((2002,2,1),(2002,1,2));
number_in_month([(2003,2,1),(2003,3,1),(2003,4,1),(2003,2,1),(2003,2,1),(2003,4,1),(2003,1,1)],2);
number_in_months([(2003,2,1),(2003,3,1),(2003,4,1),(2003,2,1),(2003,2,1),(2003,4,1),(2003,1,1)],[2,4]);
dates_in_month([(2003,2,1),(2003,3,1),(2003,4,1),(2003,2,1),(2003,2,1),(2003,4,1),(2003,1,1)],5);
dates_in_months([(2003,2,1),(2003,3,1),(2003,4,1)],[1,2,3]);
get_nth(["one","two","three"],2);
date_to_string((2003,5,27));
number_before_reaching_sum(5,[2,2,1,4,1]);
what_month(150);
month_range(150,160);
find_older([(2003,2,1),(2000,3,1),(2003,4,1),(2003,2,1),(2003,2,1),(2003,4,1),(2003,1,1)]);
find_older([]);
```

Результат

```
> val it = false: bool;
> val it = 3: int;
> val it = 5: int;
> val it = [(2003, 5, 1), (2003, 5, 1), (2003, 5, 1), (2003, 5, 1), (2003, 5, 1), (2003, 5, 1), (2003, 5, 1)]: (int * int * int) list;
> val it = [[(2003, 1, 1), (2003, 1, 1), (2003, 1, 1)], [(2003, 2, 1), (2003, 2, …), (2003, …)], [(2003, 3, 1), (2003, 3, …), (2003, …)]]: (int * int * int) list list;
> val it = "two": string;
> val it = "May 27,2003": string;
> val it = 2: int;
> val it = 5: int;
> val it = [5, 5, 6, 6, 6, 6, 6, 6, 6, 6]: int list;
> val it = SOME (2000, 3, 1): (int * int * int) option;
> val it = NONE: (int * int * int) option;
```