1. **Cząsteczki 2D** (C++ i Java)

Zadaniem będzie stworzenie konsolowego symulatora zachowania cząsteczek w przestrzeni dwuwymiarowej. Każda z rodzajów cząsteczek powinna przechowywać informacje o swoim położeniu $A_t = (x,y)$ i każda będzie się przemieszczać w danym kierunku określonym zwrotem $v_t = [v_x, v_y]$ oraz promieniu r_t ($r_t \in (0, 0.5)$). Zakładamy jednocześnie że cząsteczki to idealne kule. Cząsteczki posiadać będą również informacje o skalarnej masie m_t (zakładamy że $m_t \in (0, 1)$ dla prostoty symulacji). Masa powinna działać lokalnie w obrębie 2 jednostek długości i powinna przyciągać zgodnie ze wzorem:

$$\Delta v = max\{2 - |k|, 0\} \cdot m_1 \cdot m_2 \cdot k$$

gdzie k to wektor pomiędzy oddziałującymi ze sobą cząsteczkami, a Δv zmiana zwrotu danej cząstki dodawana do v_t ($v_{t+1} = v_t + \Delta v$). Dla uproszczenia zakładamy, że zmiany te odbywają się w dyskretnych momentach czasowych, po których następuje przesunięcie cząsteczek o wektor v_t, tj. $A_{t+1} = A_t + v_{t+1}$.

Jeśli cząsteczki zajdą na siebie promieniami zakładamy że dochodzi do zderzenia cząsteczek, w wyniku którego powinna nastąpić losowa rekombinacja ich mas przy jednoczesnym założeniu zasady zachowania masy oraz zgodnie z ograniczeniami na masę każdej cząsteczki (przy czym powinna być możliwość osiągnięcia przez cząsteczki zerowej masy, w przypadku czego zostają one unicestwiane). Po dokonaniu odbicia obie cząsteczki powinny mieć przeciwny zwrot wektora przemieszczenia. W szczególnych przypadkach może się zdarzyć że zderzenie z rekombinacją mas zajdzie dla więcej niż dwóch cząsteczek jednocześnie. Wówczas zwrot wektorów przemieszczenia powinien zostać odwrócony tylko jednen raz.

Niektóre cząsteczki (będziemy je nazywać dualnymi) po zderzeniu z innymi powinny oprócz rekombinacji masy spowodować rekombinację promieni cząsteczek.

Niektóre cząsteczki po zderzeniu powinny zachowywać się inaczej niż pozostałe. Będziemy nazywać je cząsteczkami rozszczepialnymi. Ich zachowanie powinno być następujące. W momencie zderzenia powinny powstać dwie cząsteczki o połowie masy wynikającej z cząsteczki pierwotnej (po uwzględnieniu rekombinacji z drugą cząstęczką z którą nastąpiło zderzenie) a zwrot powstałych cząsteczek powinien być względem cząsteczki pierwotnej pod kątem prostym a względem siebie o tym samym kierunku ale przeciwnym zwrocie.

Niektóre cząsteczki, nazwijmy je fotonami, powinny nie przechowywać informację o masie i nie wpływać na działanie innych cząsteczek ale móc odbijać się od innych cząsteczek jeśli zdajdą się na ich drodze.

Symulator powinien być zdolny do generowania startowej pozycji symulacji lub też przyjmować plik wejściowy z taką pozycją w następującym formacie:

```
X Y t
T1 x1 y1 vx1 vy1 r1 m1
T2 x2 y2 vx2 vy2 r2 m2
…
```

Gdzie `X Y` to szerokość i wysokość symulatora od której cząsteczki powinny się odbijać bez utraty masy, a `t` to ilość iteracji, które symulator powinien przeprowadzić. `T1` to typ pierwszej cząsteczki (`normal`, `dual`, `fissile`, `dual_fissile` lub `photonic`). `x1 y1` to pozycje startowe pierwszej cząsteczki, `vx1 vy1` to początkowe współrzędne zwrotu przemieszczenia a `r1 m1` to początkowy promień i masa cząsteczki. Oczywiście w przypadku fotonów masa może być podana ale powinna być pomijana.

Wyjściem symulatora powinien być tor poruszania się cząsteczek z informacją o ich typie, masie i promieniu.

```
t1
E1 T1 x1 y1 r1 m1
E2 T2 x2 y2 r2 m2
…
k1
E1 E2
E4 E5 E7
…
t2
E1 T1 x1 y1 r1 m1
E2 T2 x2 y2 r2 m2
…
```

gdzie `E1`, `E2`, ... to unikalne dla cząsteczek etykiety nadane automatycznie przez symulator zgodne z wyrażeniem regularnym /^p[0-9]+$/

a `k1`, `k2` mówią o kolizjach cząsteczek w danym momencie czasowym

W języku Java należy stworzyć GUI do symulatora wczytujące plik z torem poruszania się cząsteczek i wyświetlający graficznie wynik symulacji. GUI powinno zapewniać możliwość wyświetlania szczegółowych informacji o cząsteczce na którą naciśniemy kursorem oraz możliwość wybrania momentu czasowego do którego chcemy przeskoczyć w podglądzie symulacji.

2. **Kalkulator dla liczb o zadanej precyzji** (C++) + **Arkusz kalkulacyjny** (Java)

Zadaniem jest stworzenie konsolowego kalkulatora, który potrafi wykonywać proste działania (dodawanie, odejmowanie, mnożenie, dzielenie) dla liczb nieograniczonych zakresem liczb całkowitoliczbowych (integer, long, long long) ani zmiennoprzecinkowym (float, double). Kalkulator powinien parsować podane na wejściu wyrażenie, potrafić określić żądaną przez użytkownika dokładność i wykonywać operacje zgodnie z kolejnością wykonywania działań.

Symbole prostych działań obsługiwanych przez kalkulator:

- + dodawanie
- - odejmowanie
- * mnożenie
- / dzielenie
- () nawiasy - zmieniające kolejność wykonywania działań
- : określenie dokładności/precyzji operacji

Np.

(123456789*123456789):10 lub
123456789 *:10 123456789
mówi iż należy wykonać mnożenie liczb 123456789 przez 123456789 ale tylko pierwszych 10 cyfr wyniku działania będzie znaczące (resztę należy zastąpić zerami!)

Drugi przykład

(1/3) : 0.10 lub
1 /:0.10 3
mówi iż chcemy podzielić 1 przez 3 ale wynik jaki chcemy otrzymać chcemy otrzymać z precyzją 10 miejsc po przecinku.

Kalkulator powinien obsługiwać określenie znaku liczby np. 2 - (-10) tj. dwa odjąć minus dziesięć, gdzie minus jest znakiem liczby 10, w tym też znaku + (plus).

Dodatkowo kalkulator powinien obsługiwać możliwość wyekstrahowania n-tej najmniej znaczącej cyfry dziesiętnej liczby za pomocą operatora [ ]

Np.

123456789[1] -> 9
(1+20+300)[3] -> 3
32[1]*23[1] -> 6
123456789[10] -> 0

Oraz cyfr za przecinkiem za pomocą tego samego operatora gdy poda się w zapisie liczbę ujemną.

Np.

1.0101[-2] -> 1
23 [-1] -> 0

Powinna również istnieć możliwość wyekstrahowania dowolnego podciągu cyfr danej liczby.

Np.
123456789[3..1] -> 789
10.10101[1..-1] -> 0.1

Ale np.
123456789[3..4] -> wyjątek
10.10101[1..0] -> wyjątek

Oczywiście np:
123456789[4..2] - 123456789[9] -> 677

Wyjątki powinny być czytelnie wypisywane w ramach stderr. Powinny zawierać informacje o tym przy jakim działaniu nastąpiła sytuacja wyjątkowa i z jakiego powodu ona zaszła.

Arkusz kalkulacyjny powinien wykorzystywać wcześniej stworzony konsolowy kalkulator(!). Zarówno kolumny jak i wiersze powinny być indeksowane liczbami celem ujednolicenia adresacji. I tak np. adres komórki w pierwszym rzędzie w drugiej kolumnie to $(2,1).
Adres bezwzględny powinien być określany przez kwadratowe nawiasy np. $(2,1] oznaczałoby że po skopiowaniu/przeciągnięciu formuły do komórki niżej wciąż formuła wskazywałaby na adres $(2,1] ale na przykład po skopiowaniu/przeciągnięciu formuły o jedną komórkę w prawo formuła wskazywałaby na adres $(3,1].

Arkusz powinien pozwalać na zagnieżdżanie adresowania tj. powinno być dozwolone np. $($(1,1), 1).

Zapis zakresów powinien być dokonywany przez dwie kropki np. formuła:

=SUM($(1..2, 1))

powinna posumować liczby z pierwszego wiersza w kolumnie pierwszej i drugiej.

Również tutaj powinien działać operator określenia dokładności/precyzji operacji oraz operator ekstrahowania cyfr.

np.:
=AVG($(1, 1..10)):0.3 powinno zwrócić średnią z dziesięciu komórek w pierwszej kolumnie z precyzją do 3 miejsca po przecinku
=123456789[1] powinno zwrócić 9.

Funkcje, które powinien wspierać arkusz kalkulacyjny: SUM, AVG, STD_DEV, VARIANCE. Oczywiście arkusz powinien wspierać również pozostałe proste działania matematyczne takie jak dodawanie odejmowanie mnożenie i dzielenie.

Trudniejsze formuły które mogą być weryfikowane w trakcie sprawdzania projektu:

=SUM($(1..$(1,1),5))
=($(1,1) * $[2,2)[4..2]):20
=SUM($(1..10,1..10)) -> suma wszystkich elementów po dwóch wymiarach