# Hard Constraint :

1. no student attends more than one event at the same time;
2. the room is big enough for all the attending students and satisfies all the features required by the event;
3. only one event is put into each room in any timeslot;
4. events are only assigned to timeslots that are pre-defined as available for those events;
5. where specified,  events are scheduled to occur in the correct order in the week

## Pseudocode HC 1 :

Membuat Conflict Matrix untuk setiap event, event apa yang setiap siswa ambil (ReadFile Line 185-202)

|  |
| --- |
| **procedure** Creating ConflictMatrix  **for** every student i to student.length do  **for** event i=0 do  **for** event j =0 do  eventi= event(i)  eventj=event(j)  ConflictMatrix[eventi][eventj]++  ConflictMatrix[eventi][eventj]++  **end for**  **end for**  **end for**  **end procedure** |

Setelah membuat ConflictMatrix, diurutkan ConflictCourse untuk mengetahui event yang memiliki konflik. ConflictCourse didapat dari ConflictMatrix sehingga didapatkan event i bersamaan dengan event apa saja (ReadFile line 205-232)

|  |
| --- |
| **procedure** Creating ConflictCourse  arrayList temp 🡨 0  **for** ConflictMatrix  **for** i=0 to event.length do  **for** j =0 to conflictMatrix.length do  if ConflictMatrix[i][j]>=1 dan i!=j  Then Add temp=j  **end if**  **end for**  add Conflictcourse[temp(i)]  **end for**  **end for**  **end procedure** |

**Hard Constraint 1** (InitialSolution Line 44-49, 53, 74-75, 87)

|  |
| --- |
| **procedure** HC1  sort ConflictCourse from Largest to Smallest  Int index 🡨 ConflictCourse  Int CourseTimeslot  **For** index i =0 to conflictCourse.length do  **For** currentSlot j =0  **For** ConflictCourse k=0  Int numCourse 🡨 ConflictCourse[i][k] -> **Mengecek satu2, event yang konflik dengan event index**  If CourseTimeSlot[numCourse]=currentSlot -> **Pada Timeslot event yang konflik tersebut memiliki slot yang sama dengan event index yang di awal**  Then Return False  Else Return True  courseTimeslot[index] = j+1 // event ditaruh ditimeslot tersebut  **End If**  **End For**  **End For**  **End For**  **End Procedure** |

## Pseudocode HC 2 :

Membuat matrix sebanyak room x features (Readfile SuitableRoom)

HC 2 (Initial Solution 44-61, 98)

|  |
| --- |
| **Procedure** HardConstraint2  **For** j =0 to timeslot.length do  If HC1 True  **For** SuitableRoom k = 0  Int sr = SuitableRoom[j][k] // **Mengecek matrix suitable rooms apakah memenuhi atau tidak**  If sr == 1  Then courseTimeslot[index] = j+1 // **event ditaruh ditimeslot tersebut**  courseRoom[index] = k+1; // **memberi tanda bahwa ruangan yang digunakan untuk event ke index pada slot k+1**  **End if**  **End For**  **End if**  **End For** |

## Pseudocode HC 3

Membuat matrix sebanyak timeslot x ruangan yang berisi nilai 0 (ReadFile TimeslotRoom)

HC 3 (Initial Solution 44-61, 98)

|  |
| --- |
| **Procedure** HardConstraint3  **For** j =0 to timeslot.length do  If HC1 and HC2 True  **For** TimeslotRoom k = 0  Int tr = TimeslotRoom[j][k]  If tr == 0  Then courseTimeslot[index] = j+1 // event ditaruh ditimeslot tersebut  Then TimeslotRoom[j][k]++ // memberi nilai 1 pada rooms di slot tersebut yang berarti terisi  **End if**  **End For**  **End if**  **End For** |

## Pseudocode HC 4

Membuat Matrix untuk Event dan Slot yang diperbolehkan (ReadFile SuitableSlot)

HC 4 (Initial Solution 44-61, 70-92)

|  |
| --- |
| **procedure** HC4  sort ConflictCourse from Largest to Smallest  Int index 🡨 ConflictCourse  Int CourseTimeslot  **For** index i =0 to conflictCourse.length do  **For** currentSlot j =0  **For** ConflictCourse k=0  Int SlotAccept 🡨 Suitableslot[i][j] -> **Mengecek apakah slot pada event I dengan timeslot j bernilai 1**  If SlotAccept!=1  Then Return False  Else Return True  courseTimeslot[index] = j+1 // event ditaruh ditimeslot tersebut  **End If**  **End For**  **End For**  **End For**  **End Procedure** |

## Pseudocode HC 5

Membuat Matrix untuk Event dan EventOrder yang ditentukan (ReadFile SuitableOrder)

HC 2 (Initial Solution 44-61, 95-114)

|  |
| --- |
| **Procedure** HardConstraint5  Int index 🡨 ConflictCourse  **For** j =0 to timeslot.length do  If HC1 and HC2 and HC3 and HC4 True  Int CurrentTimeSlot  **For** CourseTimeSlot i = 0  If CourseTimeSlot[i] > 0 //Mengecek di CourseTimeslot, Event ke i yang telah terisi slot-nya  Int cekSlot = CourseTimeSlot[i] // mengambil nilai slot I pada courseTimeSlot  If CurrentTimeSlot<CekSlot && SuitableOrder[index][i] !=1 // Apabila matrix event index dan i dibandingkan current timeslot dan nilai slot event I tidak sama dengan 1  Then Return False  Else If CurrentTimeSlot>CekSlot && SuitableOrder[index][i] ==1 // Apabila matrix event index dan i dibandingkan current timeslot dan nilai slot event I tidak sama dengan 1  Then Return False  **End if**  **End For**  **End if**  **End For** |