```
    ried out. If pren = 0, then there are no preassignments;
  course := preassign [j,1]; time:= preassign [j,2];
  comment We now attempt to assign this course to the given
    time;
SCRUTINIZE: if row [course] < 0 then
  begin outstring (1, 'This course'); outinteger (1, course);
    outstring (1, 'is already scheduled at time');
    outinteger (1, -row[course]);  go to NEXT
  end;
  if number [time] + w[course] > max then
  begin outstring (1, 'Space is not available for course');
    outinteger (1, course);  outstring (1, 'at  time');
    outinteger (1, time);  go to NEXT
  end;
  for i := 1 step 1 until m do
    if row [i] = - time then
    begin if incidence [i, course] then
      begin outstring (1, 'course  number');
        outinteger (1, course);  outstring (1, 'conflicts with');
        outinteger (1,i);
        outstring (1, 'which is already scheduled at');
        outinteger (1, time),
        go to NEXT
      end if incidence
    end if row;
SATISFACTORY: row[course] := -time;
  number [time] := number [time] + w [course];
  preset := true;
NEXT:
  end THE PREASSIGNMENT;
MAIN PROGRAM: begin Boolean array available [1:m];
  integer next;
  procedure check (course);  integer course;
  begin integer j;  comment  This procedure renders un-
      available those courses conflicting with the given course;
    for j := 1 step 1 until m do
      if incidence [course,j] then available [j] := false
  end of procedure check.
    For each of the n time periods we select a suitable set of non-
    conflicting courses whose students will fit the examination
    room;
START OF MAIN PROGRAM:
  for time := 1 step 1 until n do
    if preset ≡ number[time] > 0 then
    begin comment  The preceding Boolean equivalence di-
        rects the attention of the program initially only to
        those times where prescheduling has occurred. We now
        determine the available courses (i.e. unscheduled and
        nonconflicting). If course i is already scheduled, then
        row[i] is negative;
      completed := true;
      for i := 1 step 1 until m do if row [i] > 0 then
      begin available [i] := true; completed := false end
        else available [i] := false;
      if completed then go to OUTPUT;
      if preset then
      begin comment  Some courses were prescheduled at
          this time. It is necessary to render their conflicts un-
          available;
        for i := 1 step 1 until m do
          if row[i] = -time then check (i)
      end prescheduled courses.
        We now select the available course with the most con-
        flicts. This is essentially the heuristic step and there-
        fore the place where variations on the method may be
        made;
AGAIN:
      sum := 0;
      for i := 1 step 1 until m do
        if available [i] ∧ row [i] > sum then
        begin next := i;  sum := row [i] end most conflicts;
      if sum > 0 then
      begin comment  There exists an available course, so
          we test it (viz next) for size. If it does not fit we look
          for another;
        available [next] := false;
        if number [time] + w[next] > max then go to AGAIN;
        comment  If we are here the course will fit so we use it;
        row [next] := -time;
        number [time] := number [time] + w[next];
        check (next); go to AGAIN
      end sum > 0
    end of the time loop;
    if preset then
      begin preset := false; go to START OF MAIN
        PROGRAM end
        In case of prescheduling this takes us back to try the re-
        maining time periods.
          If we have reached here with completed true then all
        courses are scheduled, but the converse may not be true,
        therefore;
    if ¬ completed then
    begin completed := true;
      for i := 1 step 1 until m do
        if row [i] > 0 then completed := false
    end ¬ completed and
  end of the main program;
OUTPUT: if ¬ completed then
  begin comment  The following for statement outputs the
      courses that were not scheduled;
    outstring (1, 'courses not scheduled');
    for i := 1 step 1 until m do
      if row [i] > 0 then outinteger (1,i)
  end not scheduled.
    The following outputs the time period j, the number of stu-
    dents number[j] and the courses i written at time j;
TIMETABLE: outstring(1, 'time  enrolment  courses');
  for j := 1 step 1 until n do
  begin outinteger (1,j); outinieger (1, number[j]);
    for i := 1 step 1 until m do
      if row[i] = -j then outinteger (1,i)
  end j.
    The following outputs the courses, the times at which they are
    written, and their enrolment;
  outstring (1, 'course  time  enrolment');
  for i := 1 step 1 until m do
    if row [i]< 0 then outinteger (1, i); outinteger (1, row [i]);
      outinteger (1, w[i])
    else
    begin outinteger(1,i); outstring(1, 'unscheduled');
      outinteger (1, w[i])
    end
end of the procedure
```

REMARK ON ALGORITHM 279
CHEBYSHEV QUADRATURE [D1]
F. R. A. Hopgood and C. Litherland
[*Comm. ACM 9* (Apr. 1966), 270]

The 33rd line of the second column on page 270 should read:

if $m \neq 4 \wedge m \neq mmax \wedge r \geq m - 4$ **then**

A printing error showed $\wedge$ as 7433.