Appendix B Programs used in Chapter Five

The following programs are written in Borland Turbo Pascal for PC Windows.
Lodge1 calculates the ploughing length, supported force and meltout associated with variable sized clasts and variable residual strengths for the till. Lodge2 takes the geometry of the features at Criccieth and calculates the potential residual strengths for the tills and potential ice velocities.

```
program lodge1;
uses WinCrt;                                  {Allows the program to draw windows}

const
 densityice = 900; {kg per m cubed}
 gravity = 9.81; {m s-1 s-1}

var
 filename : string;
 outfile : text;
 time : integer;
 temp : char;                                 {Number of model iterations of one hour}
 continuing : boolean;                        {Continuing to plough}


 {physical parameters of boulder}
 clastwidth : real;
 clastheight : real;
 conductivity : real;                         {Thermal conductivity of the rock}
 clasttillcontactheight : extended;
 clasticecontactheight : extended;

 {physical parameters of till}
 residualstrength : extended;
 internalfriction : real;
 bedslope : real; {in radians}                {Local bedslope in radians}

 {lodgement parameters}
 movementtotal : extended;                    {Total ploughing path}
 movedistance : extended;                     {Distance moved so far}
 strain : extended; {for this unit time}
 stresstransfered : extended;
 forcefromresidualstrength : extended;
 shearstress : extended;
 dragwhenlodged : extended;
```

```
 fill : extended;                          {Inflow of till from infront}
 coeffofheight : extended;
 volume : extended;
 oldfillheight : extended;
 volumetotal : extended;

 {ice parameters}
 icevelocity : real;
 debrisfraction : real; {vol of debris / vol of ice}

{----------------------procedures and functions-----------------------------}

{----------------------read in variables-----------------------------------}

procedure entervars;

begin
{physical parameters of boulder}
writeln ('Enter clast width (= depth downice also) ');
readln (clastwidth);
writeln ('Enter clast height ');
readln (clastheight);

{physical parameters of till and ice }
writeln ('Enter till residual strength ');
readln (residualstrength);

{set parameters}

conductivity := 429.92244;{per hour}   {0.1194229 joules per degree C per sec
                                          per cm, then divided by 100 for path
                                          length increase from cm to m,
                                          and x 10000 for area increase }

{physical parameters of till and ice}
internalfriction := 22; {in degrees - converted to radians later}
bedslope := 5; {in degrees - converted to radians later}
{ice parameters}
icevelocity := 0.002276867;{per hour} {0.00000063246m/s} {20m a-1}
debrisfraction := 0.20; {vol of debris / vol of basal ice}


{convert degrees to radians}
bedslope := bedslope * (Pi/180);
iceslope := iceslope * (Pi/180);
internalfriction := internalfriction * (Pi/180);

end;
```

```
{-----------------------calculate drag when lodged----------------------------}

procedure calcdrag;                    {Calculated from the Weertman, 1957, equations}

begin
 if (clastwidth=0.01) then dragwhenlodged := 214.7;
 if (clastwidth=0.1) then dragwhenlodged := 2147.98;
 if (clastwidth=1.75) then dragwhenlodged := 37583.037;
end;

{-----------------------check for lodgement-----------------------------------}

function lodged : boolean;

begin
 if (forcefromresidualstrength >= dragwhenlodged) or (clasticecontactheight <= 0) then
  begin
   lodged := true;
   writeln ('lodged')
  end
 else
  begin
   lodged := false;
  end;
end;



{--------------------calc shear stress melt and creep-------------------------}

function stresseffect (stress : extended) : extended;

const
 C = 0.0742; {degrees Celsius per Pa}
 heatoffusion = 333333.333; {joules per kg}
 heatloss = 1; {proportion}
 B = 1.98e-20; {per hour, 5.5e-15 s-1 kpa-1 from Glen's law.
                Value from Paterson, 1981, table 3.1 zero Celsius}
 n = 3; {from Glen's law}

var
 actiondistance : extended;            {Weertman's distance of creep action}
 meltdistance : extended;
 creepdistance : extended;
```

```
begin

 meltdistance := ((heatloss*C*stress*conductivity)/(heatoffusion*densityice*clastwidth));
 actiondistance := clastwidth; {as used by Weertman}
 creepdistance := ((2/9)*B*actiondistance*(exp((n)*ln(stress))));
 stresseffect := (meltdistance + creepdistance);

end;

{---------------------------clast movement-----------------------------------}

procedure calcclastmovement;

begin

 movedistance := (icevelocity -
                  (stresseffect (stresstransfered)));
 if (movedistance < 0) then movedistance := 0;
 movementtotal := movementtotal + movedistance;

end;


{------------------calc meltout sediment volume-----------------------------}

function fillheight : extended;

const
 C = 0.0742; {degrees Celsius per Pa}
 heatoffusion = 333333.333; {joules per kg}
 heatloss = 1; {proportion}

var
 meltvolume : extended; {per unit time}
 force : extended;
 volume : real;
 sedimentlength : real;

begin

 force := forcefromresidualstrength;
 meltvolume := ((heatloss*C*force*conductivity)
                        /(heatoffusion*densityice*clastwidth));
 volume := (meltvolume * debrisfraction);
 volumetotal := volumetotal + volume;
 sedimentlength :=
```

```
 (exp((1/2)*ln((2*volume)/((sin(internalfriction))/(cos(internalfriction))))));
 oldfillheight := fill;
if (sedimentlength > movedistance)then fill := ((volume/sedimentlength)+
 ((oldfillheight*(sedimentlength-movedistance))/sedimentlength))
 else fill := (volume/sedimentlength);
fillheight := fill;

end;

{------------------calc new clast and till variables-------------------------}

procedure calcnewvars;

var
 coverdistance : extended; {distance until clast covered}
 tillinflow : extended;
 extra : extended;

begin
 coverdistance := (clastheight/((sin(bedslope))/(cos(bedslope))));
 clasttillcontactheight := clastheight - (((coverdistance)-
 (movementtotal))*((sin(bedslope))/(cos(bedslope))));


 extra := (0.5*(movedistance))*((movedistance)*((sin(bedslope))/(cos(bedslope))));
 tillinflow := (movedistance * clasttillcontactheight) + extra;
 if (movedistance <= 0) then clasticecontactheight := (clasticecontactheight - (fillheight))
                                    {accounts for lack of till inflow and dividing by zero}
 else clasticecontactheight := (clastheight - ((fillheight) + ((tillinflow)/(movedistance))));

end;

{--------------------calc force of till on clast-----------------------------}

procedure calcresidualforce;

begin

 forcefromresidualstrength := (clasttillcontactheight*clastwidth*residualstrength);
 stresstransfered := forcefromresidualstrength/(clastwidth*clasticecontactheight);

end;


{---------------------------main program-------------------------------------}

begin
```

```
repeat

entervars;                                                    {Calls function}
volumetotal := 0;
movementtotal := 0;
clasttillcontactheight := 1e-12;
clasticecontactheight := clastheight - 1e-12;
forcefromresidualstrength := 1e-12;
stresstransfered := 1e-12;
time := 0;
writeln ('Enter name of output file ');
readln (filename);
assign (outfile,filename);
rewrite (outfile);

calcdrag;
continuing := true;
fill := 0;

while not lodged and continuing do                {Calls function 'lodged'}

begin
 time := time + 1;
 write (outfile,time);
 calcdrag;{drag changes with ploughdepth}                     {Calls function}
 write (outfile,dragwhenlodged);
 calcclastmovement;                                           {Calls function}
 write (outfile,movementtotal);
 calcnewvars;                                                 {Calls function}
 calcresidualforce;                                           {Calls function}
 write (outfile,forcefromresidualstrength);
 write (outfile,stresstransfered);
 write (outfile,clasttillcontactheight);
 write (outfile,clasticecontactheight);
 writeln (outfile,fill);
 if (movedistance <= 0) then continuing := false;
end;


 close (outfile);
 writeln ('Done');
 write ('volume of sediment = ');
 writeln (volumetotal);
 writeln ('do you wish to run another test?');
 readln (temp);

until temp = 'n';
```

```
end.
```

```
program lodge2;
uses WinCrt;                                    {Allows the program to draw windows}

const
 densityice = 900; {kg per m squared}
 gravity = 9.81; {m s-1 s-1}

var
 filename : string;                             {For the name of the output file}
 outfile : text;
 time : extended;                               {Number of model iterations of one day}
 continuing : boolean;                          {Continuing to plough}

 {physical parameters of boulder}
 clastwidth : real;
 clastheight : real;
 conductivity : real;                           {Thermal conductivity of the rock}
 clasttillcontactheight : extended;
 clasticecontactheight : extended;

 {physical parameters of till}
 residualstrength : extended;
 internalfriction : real;
 bedslope : real;                                       {Local bedslope in radians}

 {lodgement parameters}
 movementtotal : extended;                              {Total ploughing path}
 movedistance : extended;                               {Distance moved so far}
 stresstransfered : extended;
 forcefromresidualstrength : extended;
 dragwhenlodged : extended;
 fill : extended;                               {Inflow of till from infront}
 coeffofheight : extended;
 volume : extended;
 oldfillheight : extended;
 volumetotal : extended;
 loopresidual : integer;
 loopicevelocity : integer;
 maxforce : extended;                           {Maximum force supported on clast}

 {ice parameters}
 icevelocity : real;
 debrisfraction : real; {vol of debris / vol of ice}
```

```
{----------------------procedures and functions----------------------------}

{----------------------read in variables-----------------------------------}

procedure entervars;

begin

 {physical parameters of boulder, till and ice}
 conductivity := 10318.13856;{per day}          {0.1194229 joules per degree C per sec per
                                                 cm, then divided by 100 for path length
                                                 increase and x 10000 for area increase }
 internalfriction := 22;
 bedslope := 5;
 debrisfraction := 0.20; {vol of debris / vol of basal ice}
 clastwidth := 1.75;
 clastheight := 1.0;

 {convert degrees to radians}
 bedslope := bedslope * (Pi/180);
 internalfriction := internalfriction * (Pi/180);

end;

{----------------------calculate drag when lodged--------------------------}

procedure calcdrag;                {Drag calculated using the Weertman, 1957, equations
                                        rearranged for force}
var
 A : extended;
 B : extended;

begin
 A := (2.66805e17*icevelocity)+(6930*(exp((1/2)*ln(3.68505e27+(1.48225e27*
 (exp((2)*ln(icevelocity))))))));
 B := (exp((1/3)*ln(A)));
 dragwhenlodged := ((500/231)*B)-(1.21523e12/B);
end;

{----------------------check for lodgement---------------------------------}

function lodged : boolean;

begin
 if (forcefromresidualstrength >= dragwhenlodged) or (clasticecontactheight <= 0) then
  begin
   lodged := true;
  end
```

```
 else
  begin
   lodged := false;
  end;
end;


{--------------------calc shear stress melt and creep-------------------------}

function stresseffect (stress : extended) : extended;
                                                    {Calculated using the Weertman, 1957,
                                                      equations}
const
 C = 0.0742; {degrees Celsius per Pa}
 heatoffusion = 333333.333; {joules per kg}
 heatloss = 1; {proportion}
 B = 4.752e-19; {per day, 5.5e-15 s-1 kpa-1 from Glen's law. Value from Paterson, 1981,
  table 3.1 zero Celsius}
 n = 3; {from Glen's law}

var
 actiondistance : extended; {Weertman's, 1957, distance of creep action}
 meltdistance : extended;
 creepdistance : extended;


begin
 meltdistance := ((heatloss*C*stress*conductivity)/(heatoffusion*densityice*clastwidth));
 actiondistance := clastwidth;                              {as used by Weertman, 1957}
 creepdistance := ((2/9)*B*actiondistance*(exp((n)*ln(stress))));
 stresseffect := (meltdistance + creepdistance);

end;

{--------------------------clast movement-----------------------------------}

procedure calcclastmovement;

var sheareffect : extended;

 begin

 movedistance := (icevelocity -
                  (stresseffect (stresstransfered)));
 if (movedistance < 0) then movedistance := 0;
 movementtotal := movementtotal + movedistance;

end;
```

```
{------------------calc meltout sediment volume-----------------------------}

function fillheight : extended;

const
 C = 0.0742; {degrees Celsius per Pa}
 heatoffusion = 333333.333; {joules per kg}
 heatloss = 1; {proportion}

var
 meltvolume : extended; {per unit time}
 force : extended;
 volume : real;
 sedimentlength : real;

begin
 force := forcefromresidualstrength;
 meltvolume := ((heatloss*C*force*conductivity)
                              /(heatoffusion*densityice*clastwidth));
 volume := (meltvolume * debrisfraction);
 volumetotal := volumetotal + volume;
 sedimentlength :=                                          {Calc. inflow distribution}
(exp((1/2)*ln((2*volume)/((sin(internalfriction))/(cos(internalfriction))))));
 oldfillheight := fill;
 if (sedimentlength > movedistance)then fill := ((volume/sedimentlength)+
  ((oldfillheight*(sedimentlength-movedistance))/sedimentlength))
 else fill := (volume/sedimentlength);
 fillheight := fill;

end;

{------------------calc new clast and till variables-------------------------}

procedure calcnewvars;

var
 coverdistance : extended; {distance until clast covered}
 tillinflow : extended;
 extra : extended;

begin
 coverdistance := (clastheight/((sin(bedslope))/(cos(bedslope))));
 clasttillcontactheight := clastheight - (((coverdistance)-
(movementtotal))*((sin(bedslope))/(cos(bedslope))));

 extra := (0.5*(movedistance))*((movedistance)*((sin(bedslope))/(cos(bedslope))));
```

```
 tillinflow := (movedistance * clasttillcontactheight) + extra;
 if (movedistance <= 0) then clasticecontactheight := (clasticecontactheight - (fillheight))
  {accounts for lack of till inflow and dividing by zero}
 else clasticecontactheight := (clastheight - ((fillheight) + ((tillinflow)/(movedistance))));

end;

{--------------------calc force of till on clast-----------------------------}

procedure calcresidualforce;

begin

 forcefromresidualstrength := (clasttillcontactheight*clastwidth*residualstrength);
 if (forcefromresidualstrength > maxforce) then maxforce := forcefromresidualstrength;
 stresstransfered := forcefromresidualstrength/(clastwidth*clasticecontactheight);

end;


{---------------------------main program-----------------------------------}

begin

 filename := 'testres.dat';
 assign (outfile,filename);
 rewrite (outfile);
 writeln ('started');

 icevelocity := 0;

 for loopicevelocity := 1 to 200 do
  begin
   residualstrength := 0;
   icevelocity := (((icevelocity*366) + 10)/366);{per day in 10 m pa steps}
    for loopresidual := 1 to 50 do
      begin
        residualstrength := (residualstrength + 1000);
        entervars;                                                    {Calls function}
        volumetotal := 0;
        maxforce := 0;
        clasttillcontactheight := 1e-12;
        clasticecontactheight := clastheight - 1e-12;
        forcefromresidualstrength := 1e-12;
        stresstransfered := 1e-12;
        time := 0;
        movementtotal := 0;
        oldfillheight := 0;
```

```
      calcdrag;                                                    {Calls function}
      continuing := true;
      fill := 0;

       while not lodged and continuing do

        begin
          time := time + 1;
          calcdrag;{drag changes with ploughdepth}                 {Calls function}
          calcclastmovement;                                       {Calls function}
          calcnewvars;                                             {Calls function}
          calcresidualforce;                                       {Calls function}
          if (movedistance <= 0) then continuing := false;
        end;


       if (movementtotal > 10) and (movementtotal < 11) then
        begin
          icevelocity := (icevelocity*366);
          write (outfile,icevelocity);
          icevelocity := (icevelocity/366);
          write (outfile,residualstrength);
          write (outfile,movementtotal);
          write (outfile,volumetotal);
          writeln (outfile,maxforce);
        end;


      end;
    end;
  close (outfile);
 writeln ('finished');
end.
```