Priorities:  
1) ModifierConvert - Anthony  
2) StrvTgh - Dan  
3) #of Attacks - Dan

4) ToHitFunc - Dan

5) ToWoundFunc - Winson

6) DamageFunc - Dan

7) RandomDamge - Winson

Global variables

Double #ofAttacks //holds the initial number of attacks, initialized as a double for ease of use

Double #ofHits //holds the number of hits

Double #ofWounds //holds the number of wounds

Double Damage //holds the damage

Global Functions

#of Attacks //takes the input number of attacks

ToHitFunc // Attacks to hits

ToWoundFunc // Hits to wounds

DamageFunc // Wounds to Damage

ModiferConvert // converts rolls into decimal numbers to be used to modify the running total

RandomDamge //converts the random damage into non random values, based on averages

StrvTgh //compares the Strength to Toughness to get the WoundRoll

// basic input function, starts the process of the calculator

#ofAttacks Func

cin>> #ofAttacks; // #ofAttacks is a global variable

ToHitFunc //takes Attacks and gets hits

#ofHits = #ofAttacks; // takes the number of attacks and assigned it to hits

Double skill; //Skill takes the input BS or WS, to be converted to decimal point

Bool +1 = F;

Bool +1 = F;

Bool reRoll1 = F;

cin>> Skill;

If +1 = True;

Skill -1;

If -1 = True;

Skill +1;

ModifierConvert (Skill); //ModiferConvert is a function used more than once

If reRoll1 = T;

Skill + .16;

#ofHits = #ofHits x Skill;

ToWoundFunc //takes Hits and gets wounds

#ofWounds = #ofHits; //takes the number of Hits from previous step

Int Str // weapons strength

Int Tgh // targets toughness

Double WoundRoll; //holds the value for the wound roll, gets from StrvTgh

Bool +1 = F;

Bool -1 = F;

Bool reRoll1 = F;

cin>> Str;

cin>> Tgh;

StrvTgh (Str, Tgh) //comparison function, separate call gets Wound roll

If +1 = True;

WoundRoll -1;

If -1 = True;

WoundRolll +1;

ModifierConvert (WoundRoll); //ModiferConvert is a function used more than once

If reRoll1 = T;

WoundRoll + .16;

#ofWounds = #ofHits x WoundRoll;

DamageFunc //calculates final damage

Double Damage; //holds the damage value of the weapon

Bool RandomDamage = F; //determines if the RandomDamage Func is needed

Bool InvulnPres = F; //Checks for the presence of an Invuln save

Double ArmPen; //Weapons AP stat

Double ArmSav; // Targets Armor save

Double Invuln = 100; // Targets Invuln save, set to 100 so ArmSav is always less than invuln. Users can leave this blank and the func still work

Double FinSav; //the final save to be used.

Bool DoesHavFNP = F; // Determine if Target has FNP

Double FNP = 1; // value of FNP, set as 1 for the final calculator step

cin>> RandomDamage; //not sure how, checks if the damage is random

If RandomDamage = T;

RandomDamage //separate function, outputs to Damage

RandomDamage = Damage.

else ;

cin>>Damage;

cin>>ArmPen;

cin>>ArmSav;

cin>> InvulnPres;

If InvulnPres = True;

cin>> Invuln;

ArmPen x -1;

ArmSav + ArmPen;

If ArmSav < Invuln;

ModifierConvert (ArmSav);

FinSav = ArmSav;

#ofWounds x FinSav = Result

Else;

ModifierConvert (Invuln)

FinSav = Invuln;

#ofWounds x FinSav = Result

cin>> DoesHavFNP;

If DoesHavFNP = T;

cin>> FNP;

Modifier Convert (FNP);

Result x FNP;

Else;

Return result;

ModifierConvert //converts the input whole numbers into modifiers for use in the stat calculation

Switch (Double S/W); //either skill or woundroll

Double mod; //assists in the switch

Mod = S/W.

Case (1);

Mod = .83

Break;

Case (2);

Mod = .83

Break;

Case (3);

Mod = .66

Break;

Case (4);

Mod = .5

Break;

Case (5);

Mod = .33

Break;

Case (6);

Mod = .16

Break;

Case (7);

Mod = .16

Break;

S/W = Mod;

StrvTgh (int, int); // compares S v T

If (S = T);

WoundRoll = 4;

Else if (S\*2 > T)

WoundRoll = 2;

Else if (S > T)

WoundRoll = 3;

Else if (S < T\*2)

WoundRoll = 6;

Else if (S < T)

WoundRoll = 5;

DamFunc

Take input from drop downs

If xdx run a conversion to solid number

Drop Down input 1+ input 2, assign to damage

If input 1+input 2 = 0, damage = 1