**Operating Systems Lab #3**

**PART C**

The following output was generated by running the simulation multiple times, and inputting varying numbers of Phaser Probes.

**CoptionOutput.txt**

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 1

Phaser Probes Destroyed: 1

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 18

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 2

Phaser Probes Destroyed: 2

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 21

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 3

Phaser Probes Destroyed: 3

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 21

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 4

Phaser Probes Destroyed: 4

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 25

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 5

Phaser Probes Destroyed: 5

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 33

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 6

Phaser Probes Destroyed: 6

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 27

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 7

Phaser Probes Destroyed: 7

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 50

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 8

Phaser Probes Destroyed: 8

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 51

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 9

Phaser Probes Destroyed: 9

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 47

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 10

Phaser Probes Destroyed: 10

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 40

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 11

Phaser Probes Destroyed: 11

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 4

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 12

Phaser Probes Destroyed: 12

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 0

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 13

Phaser Probes Destroyed: 11

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 0

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 14

Phaser Probes Destroyed: 10

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 1

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 15

Phaser Probes Destroyed: 5

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 1

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 2

**Simulation Statistics for Part C**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of Phaser Probes** | **Phaser Probes Destroyed** | **Torpedo Probes Destroyed** | **Asteroids Destroyed** | **Shield Damage** | **Result** |
| 1 | 1 | 2 | 18 | 5 | FAILURE |
| 2 | 2 | 2 | 21 | 5 | FAILURE |
| 3 | 3 | 2 | 21 | 5 | FAILURE |
| 4 | 4 | 2 | 25 | 5 | FAILURE |
| 5 | 5 | 2 | 33 | 5 | FAILURE |
| 6 | 6 | 2 | 27 | 5 | FAILURE |
| 7 | 7 | 2 | 50 | 5 | FAILURE |
| 8 | 8 | 2 | 51 | 5 | FAILURE |
| 9 | 9 | 2 | 47 | 5 | FAILURE |
| 10 | 10 | 2 | 40 | 5 | FAILURE |
| 11 | 11 | 2 | 55 | 4 | SUCCESS |
| 12 | 12 | 2 | 55 | 0 | SUCCESS |
| 13 | 11 | 2 | 55 | 0 | SUCCESS |
| 14 | 10 | 2 | 55 | 1 | SUCCESS |
| 15 | 5 | 1 | 55 | 2 | SUCCESS |

**Part C Discussion**

Based on the output of the C option, the turning point where the minimum amount of probes have been created for a successful journey is around twelve. Even though the simulation produced a successful traversal at eleven probes, by the end of it all eleven Phaser probes and both Torpedo probes had rammed into asteroids and the shields were one hit away from being completely destroyed. We can gather then that eleven probes sits right on the edge of being successful. On the other hand, even though the twelve probe simulation also resulted in complete destruction of the defensive probes, by the end the shields were still at full capacity, giving the fleet some buffer room to absorb asteroid hits before they completed their journey. Past the twelve Phaser Probe mark, all simulations resulted in remaining defensive probes at the end, with minimal to no shield damage. One would almost expect this result as at the twelve probe mark, there are fourteen total defensive probes available, almost one for every asteroid that can be stored.   
 At the fifteen Phaser Probe level there is a sudden drop in the number of defensive probes being destroyed; only five Phaser Probes were destroyed and a Torpedo Probe actually survived until the end. This is most likely because at that point there are seventeen total defensive probes; this means not only is there at least one probe available for every asteroid stored, but there are also two buffer Phaser Probes that can fill in for destroyed probes and can be assigned asteroids in the event that a new asteroid is detected while the other fifteen probes are busy. This ability to cover a large number of probes is most likely why there is such a drop in the number of probes being destroyed.  
 One problem with using a stack is that asteroids end up being assigned to probes in order of most recently detected. This could potentially lead to a situation where asteroids that had been previously detected are stuck in the stack behind other more recent asteroids; eventually these stuck asteroids will collide with the ship’s shields as they are never destroyed. This is most likely the reason there was shield damage in the fourteen and fifteen Phaser Probe simulations despite there being more than enough probes to handle the incoming asteroids.  
 There are potential race conditions when accessing the stack; if Push and Pop are called at roughly the same time, then the ordering of these functions could produce different results. If either function is called in the middle of the other, it could potentially create an empty space between the top of the stack and the rest of the stack. To eliminate race conditions, two precautions are taken. The first is that the stack is only ever accessed inside accept blocks, so the code functions as a critical section, and second is that the stack is implemented as a protected type, which causes the Stack to act like a monitor; Push and Pop cannot be called at the same time.

**Part C Code**

The following package contains the functions used to produce random asteroids. It is also used for the B and A options of the lab.

**RandomFunctions.ads**

with Ada.Numerics.Float\_Random;

use Ada.Numerics.Float\_Random;

package RandomFunctions is

function RandomPoisson (

Mean : Float;

Gen : Generator)

return Float;

function RandomSize (

Gen : Generator)

return Integer;

function RandomImpactTime (

Gen : Generator)

return Float;

end RandomFunctions;

**RandomFunctions.adb**

with Ada.Numerics.Float\_Random, Ada.Numerics.Elementary\_Functions;

use Ada.Numerics.Float\_Random, Ada.Numerics.Elementary\_Functions;

package body RandomFunctions is

function RandomPoisson (

Mean : Float;

Gen : Generator)

return Float is

UniformNum : Float;

begin

UniformNum := Random(Gen);

if UniformNum = 0.0 then

return 0.0;

else

return Float(- Log(UniformNum) / (1.0 / Mean));

end if;

end RandomPoisson;

function RandomSize (

Gen : Generator)

return Integer is

UniformNum : Float;

begin

UniformNum := Random(Gen);

if UniformNum <= 0.2 then

return 3;

elsif UniformNum > 0.2 and UniformNum <= 0.5 then

return 6;

elsif UniformNum > 0.5 and UniformNum <= 0.7 then

return 9;

else

return 11;

end if;

end RandomSize;

function RandomImpactTime (

Gen : Generator)

return Float is

begin

return (Random(Gen) \* 15.0);

end RandomImpactTime;

end RandomFunctions;

The following package defines the Asteroid object used to store Asteroid information. It is also used for the B and A options of the lab.

**AsteroidPackage.ads**

with Ada.Calendar;

use Ada.Calendar;

package AsteroidPackage is

type Asteroid is private;

type Asteroid\_Ptr is access Asteroid;

function GetSize (

Ast : in Asteroid\_Ptr)

return Integer;

function GetID (

Ast : in Asteroid\_Ptr)

return Integer;

function GetAbsoluteImpactTime (

Ast : in Asteroid\_Ptr)

return Duration;

procedure Construct (

Ast : in Asteroid\_Ptr;

AstID : in Integer;

SizeInput : in Integer;

ImpactTime : in Duration;

FoundTime : in Duration);

function HasImpacted (

Ast : in Asteroid\_Ptr)

return Boolean;

function RemainingTimeToImpact (

Ast : in Asteroid\_Ptr)

return Duration;

procedure ToString (

Ast : in Asteroid\_Ptr);

private

type Asteroid is

record

ID : Integer;

Size : Integer;

TimeToImpact : Duration;

TimeFound : Duration;

end record;

end AsteroidPackage;

**AsteroidPackage.adb**

with Ada.Text\_IO, Ada.Float\_Text\_IO, Ada.Integer\_Text\_IO;

use Ada.Text\_IO, Ada.Float\_Text\_IO, Ada.Integer\_Text\_IO;

package body AsteroidPackage is

package Duration\_IO is new Ada.Text\_IO.Fixed\_IO(Duration);

use Duration\_IO;

procedure Construct (

Ast : in Asteroid\_Ptr;

AstID : in Integer;

SizeInput : in Integer;

ImpactTime : in Duration;

FoundTime : in Duration) is

begin

Ast.ID := AstID;

Ast.Size := SizeInput;

Ast.TimeToImpact := ImpactTime;

Ast.TimeFound := FoundTime;

end Construct;

function GetSize (

Ast : in Asteroid\_Ptr)

return Integer is

begin

return Ast.Size;

end GetSize;

function GetID (

Ast : in Asteroid\_Ptr)

return Integer is

begin

return Ast.ID;

end GetID;

function GetAbsoluteImpactTime (

Ast : in Asteroid\_Ptr)

return Duration is

begin

return Ast.TimeFound + Ast.TimeToImpact;

end GetAbsoluteImpactTime;

function HasImpacted (

Ast : in Asteroid\_Ptr)

return Boolean is

begin

return (Ast.TimeFound + Ast.TimeToImpact) <= Seconds(Clock);

end HasImpacted;

function RemainingTimeToImpact (

Ast : in Asteroid\_Ptr)

return Duration is

begin

return (Ast.TimeFound + Ast.TimeToImpact) - Seconds(Clock);

end RemainingTimeToImpact;

procedure ToString (

Ast : in Asteroid\_Ptr) is

begin

Put("(#");

Put(Ast.ID, 2);

Put(")");

Put(" Asteroid Size: ");

Put(Ast.Size, 2);

New\_Line;

Put("(#");

Put(Ast.ID, 2);

Put(")");

Put(" Time til Impact: ");

Put(Ast.TimeToImpact, 3, 3, 0);

New\_Line;

Put("(#");

Put(Ast.ID, 2);

Put(")");

Put(" Found at: ");

Put(Ast.TimeFound, 3, 3, 0);

New\_Line(2);

end ToString;

end AsteroidPackage;

The following code defines the protected stack used to store Asteroids in the C option.

**GenericProtectedStack.ads**

generic

type Item is private;

Size :

in Positive;

package GenericProtectedStack is

subtype Item\_Range is Positive range 1..Size;

type Item\_Array is array (Item\_Range) of Item;

protected type ProtectedStack is

procedure Push (

InputItem : in Item;

Success : out Boolean);

procedure Pop (

OutputItem : out Item;

Success : out Boolean);

function IsFull return Boolean;

function IsEmpty return Boolean;

private

Stack : Item\_Array;

Top : Integer := 0;

end ProtectedStack;

end GenericProtectedStack;

**GenericProtectedStack.adb**

package body GenericProtectedStack is

protected body ProtectedStack is

procedure Push (

InputItem : in Item;

Success : out Boolean) is

begin

if IsFull then

Success := False;

return;

else

Success := True;

Top := Top + 1;

Stack(Top) := InputItem;

end if;

end Push;

procedure Pop (

OutputItem : out Item;

Success : out Boolean) is

begin

if IsEmpty then

Success := False;

return;

else

Success := True;

OutputItem := Stack(Top);

Top := Top - 1;

end if;

end Pop;

function IsFull return Boolean is

begin

return Top = Size;

end IsFull;

function IsEmpty return Boolean is

begin

return Top = 0;

end IsEmpty;

end ProtectedStack;

end GenericProtectedStack;

Finally, the following code defines the simulation of the Task Force Command and the Scout/Defensive Probes, allowing for a variable number of Phaser Probes to be entered and created before beginning the simulation.

**Coption.adb**

with RandomFunctions, AsteroidPackage, Ada.Calendar, Ada.Numerics.Float\_Random, Ada.Integer\_Text\_IO, Ada.Text\_IO, Ada.Unchecked\_Deallocation, GenericProtectedStack;

use RandomFunctions, AsteroidPackage, Ada.Calendar,

Ada.Numerics.Float\_Random, Ada.Integer\_Text\_IO, Ada.Text\_IO;

procedure Coption is

--Declarations/Functions

procedure Free is

new Ada.Unchecked\_Deallocation(Asteroid, Asteroid\_Ptr);

package AsteroidStackPackage is new GenericProtectedStack(Asteroid\_Ptr,

15);

use AsteroidStackPackage;

type DefensiveProbeType is

(Torpedos,

Phasers);

function TimeToDestroy (

Size : in Integer;

ProbeType : in DefensiveProbeType)

return Duration is

begin

if ProbeType = Torpedos then

if Size = 11 then

return Duration(6.0);

elsif Size = 9 then

return Duration(3.0);

elsif Size = 6 then

return Duration(3.0);

end if;

elsif ProbeType = Phasers then

if Size = 11 then

return Duration(6.0);

elsif Size = 9 then

return Duration(4.0);

elsif Size = 6 then

return Duration(2.0);

end if;

end if;

return Duration(0.0);

end TimeToDestroy;

--Task Types

type TFC;

type DefensiveProbeArray;

type TFC\_Ptr is access all TFC;

type DPA\_Ptr is access all DefensiveProbeArray;

task type TFC is

entry Initialize (

ID : in Integer;

TFC\_ID : in TFC\_Ptr;

DPA : in DPA\_Ptr);

entry NotifyOfAsteroid (

NewTarget : in out Asteroid\_Ptr);

entry RequestTarget (

Target : out Asteroid\_Ptr;

Available : out Boolean);

entry NotifyOfTargetDestroyed;

entry NotifyOfDeath (

DPType : in DefensiveProbeType);

end TFC;

task type ScoutProbe is

entry Initialize (

TFCParent : in TFC\_Ptr);

end ScoutProbe;

task type DefensiveProbe is

entry Initialize (

TFCParent : in TFC\_Ptr;

DPType : in DefensiveProbeType;

ProbeID : in Integer);

end DefensiveProbe;

type DPL is array (Integer range <>) of DefensiveProbe;

type DefensiveProbeArray

(NumProbes : Integer := 1) is

record

Probes : DPL (1 .. NumProbes);

end record;

--Task Bodies

task body TFC is

--ID for tracking simulations

SimID : Integer;

--Data Structures

AstStack : ProtectedStack;

Scout : ScoutProbe;

TorpedoOne : DefensiveProbe;

TorpedoTwo : DefensiveProbe;

PhaserProbes : DPA\_Ptr;

--Mission Statistics

TorpedoDeaths : Integer := 0;

PhaserDeaths : Integer := 0;

Destroyed : Integer := 0;

Damage : Integer := 0;

MissionComplete : Boolean := False;

begin

accept Initialize (

ID : in Integer;

TFC\_ID : in TFC\_Ptr;

DPA : in DPA\_Ptr) do

SimID := ID;

PhaserProbes := DPA;

Scout.Initialize(TFC\_ID);

TorpedoOne.Initialize(TFC\_ID, Torpedos, 1);

TorpedoTwo.Initialize(TFC\_ID, Torpedos, 2);

for I in PhaserProbes.Probes'range loop

PhaserProbes.Probes(I).Initialize(TFC\_ID, Phasers, I+2);

end loop;

end Initialize;

while not MissionComplete loop

select

--Scout Probe Report New Asteroid

accept NotifyOfAsteroid (

NewTarget : in out Asteroid\_Ptr) do

--put\_line("Found new asteroid:");

--ToString(NewTarget);

declare

Success : Boolean;

begin

AstStack.Push(NewTarget, Success);

if not Success then

Damage := Damage + 1;

--("Destroyed Asteroid #"); put(GetID(NewTarget), 2); put(" (Shields Hit)."); new\_line;

Free(NewTarget);

Destroyed := Destroyed + 1;

end if;

end;

end NotifyOfAsteroid;

or

--Defensive Probe Request Target

accept RequestTarget (

Target : out Asteroid\_Ptr;

Available : out Boolean) do

declare

TempPtr : Asteroid\_Ptr;

Success : Boolean;

begin

GetTargetLoop:

loop

AstStack.Pop(TempPtr, Success);

if Success then

if HasImpacted(TempPtr) then

--put("Destroyed Asteroid #"); put(GetID(TempPtr), 2); put(" (Shields Hit)."); new\_line;

Damage := Damage + 1;

Free(TempPtr);

Destroyed := Destroyed + 1 ;

else

Target := TempPtr;

Available := True;

exit GetTargetLoop;

end if;

else

Available := False;

exit GetTargetLoop;

end if;

end loop GetTargetLoop;

end;

end RequestTarget;

or

--Defensive Probe Report Target Destroyed

accept NotifyOfTargetDestroyed do

Destroyed := Destroyed + 1;

end NotifyOfTargetDestroyed;

or

--Defensive Probe Report Death

accept NotifyOfDeath (

DPType : in DefensiveProbeType) do

if DPType = Torpedos then

TorpedoDeaths := TorpedoDeaths + 1;

elsif DPType = Phasers then

PhaserDeaths := PhaserDeaths + 1;

end if;

Destroyed := Destroyed + 1;

end NotifyOfDeath;

end select;

--Check for mission end

if Destroyed >= 55 then

MissionComplete := True;

New\_Line;

Put("-- Mission Stats for Simulation --");

New\_Line;

Put("Status: ");

Put("SUCCESS");

New\_Line;

Put("Total # of Phaser Probes: ");

Put(PhaserProbes.Probes'Length, 2);

New\_Line;

Put("Phaser Probes Destroyed: ");

Put(PhaserDeaths, 2);

New\_Line;

Put("Total # of Torpedo Probes: 2");

New\_Line;

Put("Torpedo Probes Destroyed: ");

Put(TorpedoDeaths, 2);

New\_Line;

Put("Total # of Asteroids Destroyed: ");

Put(Destroyed, 2);

New\_Line;

Put("Shield Damage Taken: ");

Put(Damage, 2);

New\_Line(2);

elsif Damage >= 5 then

MissionComplete := True;

New\_Line;

Put("-- Mission Stats for Simulation --");

New\_Line;

Put("Status: ");

Put("FAILURE");

New\_Line;

Put("Total # of Phaser Probes: ");

Put(PhaserProbes.Probes'Length, 2);

New\_Line;

Put("Phaser Probes Destroyed: ");

Put(PhaserDeaths, 2);

New\_Line;

Put("Total # of Torpedo Probes: 2");

New\_Line;

Put("Torpedo Probes Destroyed: ");

Put(TorpedoDeaths, 2);

New\_Line;

Put("Total # of Asteroids Destroyed: ");

Put(Destroyed, 2);

New\_Line;

Put("Shield Damage Taken: ");

Put(Damage, 2);

New\_Line(2);

end if;

end loop;

end TFC;

task body ScoutProbe is

--Data

RandomNum : Generator;

Parent : TFC\_Ptr;

--Temp storage variables

Ast : Asteroid\_Ptr;

AstNumber : Integer := 1;

--Control variable

Recalled : Boolean := False;

begin

accept Initialize (

TFCParent : in TFC\_Ptr) do

Parent := TFCParent;

--Seed generator for random output

Reset(RandomNum, Integer(Seconds(Clock)));

end Initialize;

while not Recalled loop

delay Duration(RandomPoisson(4.0, RandomNum));

Ast := new Asteroid;

Construct(Ast, AstNumber, RandomSize(RandomNum), Duration(

RandomImpactTime(RandomNum)), Seconds(Clock));

AstNumber := AstNumber + 1;

Parent.NotifyOfAsteroid(Ast);

end loop;

end ScoutProbe;

task body DefensiveProbe is

--Data

Parent : TFC\_Ptr;

ProbeType : DefensiveProbeType;

ID : Integer;

--Temp storage variables

Target : Asteroid\_Ptr;

Available : Boolean;

--Control variable

SelfDestruct : Boolean := False;

begin

accept Initialize (

TFCParent : in TFC\_Ptr;

DPType : in DefensiveProbeType;

ProbeID : in Integer) do

Parent := TFCParent;

ProbeType := DPType;

ID := ProbeID;

end Initialize;

while not SelfDestruct loop

Parent.RequestTarget(Target, Available);

if Available then

if TimeToDestroy(GetSize(Target), ProbeType) <=

RemainingTimeToImpact(Target) then

--Destroy

delay TimeToDestroy(GetSize(Target), ProbeType);

--put("Destroyed Asteroid #"); put(GetID(Target), 2); put(" (D-Probe #"); put(ID, 2); put\_line(" - Destroyed)."); new\_line;

Free(Target);

Parent.NotifyOfTargetDestroyed;

if ProbeType = Torpedos then

delay 3.0;

else

delay 2.0;

end if;

else

--Self Destruct

delay RemainingTimeToImpact(Target);

--put("Destroyed Asteroid #"); put(GetID(Target), 2); put(" (D-Probe #"); put(ID, 2); put\_line(" - Self Destruct)."); new\_line;

Free(Target);

Parent.NotifyOfDeath(ProbeType);

SelfDestruct := True;

end if;

else

delay 0.5;

end if;

end loop;

end DefensiveProbe;

Simulation : aliased TFC;

Temp : DPA\_Ptr;

NumProbes : Integer;

begin

Put("Enter # of Phaser Probes: ");

Get(NumProbes);

Temp := new DefensiveProbeArray(NumProbes);

Simulation.Initialize(1, Simulation'access, Temp);

Put\_Line("Running Simulation...");

end Coption;

**PART B**

The following output was generated by running the simulation multiple times, and inputting varying numbers of Phaser Probes.

**BoptionOutput.txt**

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 1

Phaser Probes Destroyed: 1

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 11

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 2

Phaser Probes Destroyed: 2

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 1

Total # of Asteroids Destroyed: 26

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 3

Phaser Probes Destroyed: 3

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 18

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 4

Phaser Probes Destroyed: 4

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 30

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 5

Phaser Probes Destroyed: 5

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 29

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 6

Phaser Probes Destroyed: 6

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 33

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 7

Phaser Probes Destroyed: 7

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 42

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 8

Phaser Probes Destroyed: 8

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 44

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 9

Phaser Probes Destroyed: 9

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 42

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 10

Phaser Probes Destroyed: 10

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 50

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 11

Phaser Probes Destroyed: 11

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 1

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 12

Phaser Probes Destroyed: 9

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 1

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 13

Phaser Probes Destroyed: 11

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 1

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 14

Phaser Probes Destroyed: 9

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 1

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 15

Phaser Probes Destroyed: 8

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 2

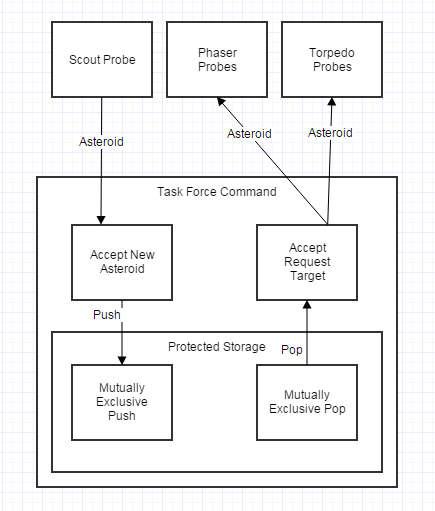
**Simulation Statistics for Part B**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of Phaser Probes** | **Phaser Probes Destroyed** | **Torpedo Probes Destroyed** | **Asteroids Destroyed** | **Shield Damage** | **Result** |
| 1 | 1 | 2 | 11 | 5 | FAILURE |
| 2 | 2 | 1 | 26 | 5 | FAILURE |
| 3 | 3 | 2 | 18 | 5 | FAILURE |
| 4 | 4 | 2 | 30 | 5 | FAILURE |
| 5 | 5 | 2 | 29 | 5 | FAILURE |
| 6 | 6 | 2 | 33 | 5 | FAILURE |
| 7 | 7 | 2 | 42 | 5 | FAILURE |
| 8 | 8 | 2 | 44 | 5 | FAILURE |
| 9 | 9 | 2 | 42 | 5 | FAILURE |
| 10 | 10 | 2 | 50 | 5 | FAILURE |
| 11 | 11 | 1 | 55 | 5 | SUCCESS |
| 12 | 9 | 2 | 55 | 1 | SUCCESS |
| 13 | 11 | 2 | 55 | 1 | SUCCESS |
| 14 | 9 | 2 | 55 | 1 | SUCCESS |
| 15 | 8 | 1 | 55 | 2 | SUCCESS |

**Part B Discussion**

As with the C option, there is again a clear turning point at around the eleven to twelve probe mark. Again, the eleven probe run resulted in a barely successful simulation; the shields were completely depleted by the end of the journey. The fact that this was a successful simulation seems to suggest that the 55th asteroid destroyed by the fleet was destroyed by a collision with the shield and not the remaining Torpedo Probe. This once more points towards the eleven probe mark as sitting somewhere on the edge of being successful. The twelve probe mark still remains the most likely threshold for the minimum number of probes, as not only was it successful, but it also resulted in three remaining probes with very minimal shield damage.   
 Although the results point to the same threshold, they are not the same. If you compare the number of asteroids destroyed, there is a clear and almost steady increase as more probes are added in the B option, whereas in the C option there is a sudden jump at around the seven probe mark. This is most likely because of the nature of the data structures used to store the asteroids in each option. In the C option, it was possible for asteroids to effectively starve at the bottom of the stack, never being serviced in time and resulting in a collision with the ship. The seven probe mark is most likely where the number of probes is great enough to nearly prevent this from ever happening. After this point, the number of asteroids destroyed seems to stay at around the forty to fifty mark, suggesting that even though asteroids aren’t starving as much, there are still not enough probes to help mitigate the loss of one probe enough for a successful simulation until around the twelve probe mark. In the B option, it is impossible for asteroids to starve in the same way. Instead every asteroid will eventually be dealt with no matter what. It is possible, though, for an asteroid to wait in line for too long and crash into the ship by the time a probe is available. When considering this fact, it makes sense that the addition of more probes results in a more linear progression of asteroids destroyed; as the number of probes increases, the rate at which asteroids are assigned increases and thus the time spent waiting in the queue by an asteroid decreases. This decrease in wait time would obviously result in more asteroids being destroyed as it directly counters what would cause them to crash into the ship.  
 One problem with this approach is that while it guarantees that all asteroids will be processed (as opposed to potentially starving), it does not prioritize the asteroids in any way; they are merely processed in the order in which they are detected. Several potential advantages go unutilized. For instance, the Torpedo Probes can more effectively deal with asteroids of 9 units than Phaser Probes, but using this approach the chance that a Torpedo Probe is assigned an asteroid of that size is entirely random. This argument also works for Phaser Probes and asteroids of size 6. This approach also places no consideration on the remaining time until impact for each asteroid. For example, if an asteroid is detected that has the potential to be dealt with if immediately assigned to a probe, but is has to wait in the queue, then most likely the asteroid will simply collide with the shield.  
 Again, the queue used is a critical resource that could result in race conditions if the Push and Pop functions are called at the same time; the internal indices used to track the front and back of the queue could potentially get placed in a configuration that makes no sense, such as the them being at different locations when the queue is empty or at the same location when the queue is not empty. All instances of queue access are placed inside accept blocks, which treats their execution as a critical section. Also, the queue is implemented as a protected type, ensuring that Push and Pop are always executed under mutual exclusion, no matter what task is calling them.

**Block Diagram**



**Part B Code**

**RandomFunctions.ads/adb** – See C option code.

**AsteroidPackage.ads/adb** – See C option code.

The following code defines a protected queue used to store asteroids in the B option.

**GenericProtectedQueue.ads**

generic

type Item is private;

Size :

in Positive;

package GenericProtectedQueue is

subtype Item\_Range is Integer range 0..(Size-1);

type Item\_Array is array (Item\_Range) of Item;

protected type ProtectedQueue is

procedure Push (

InputItem : in Item;

Success : out Boolean);

procedure Pop (

OutputItem : out Item;

Success : out Boolean);

function IsFull return Boolean;

function IsEmpty return Boolean;

private

Queue : Item\_Array;

First : Integer := 0;

Last : Integer := 0;

Count : Integer := 0;

end ProtectedQueue;

end GenericProtectedQueue;

**GenericProtectedQueue.adb**

package body GenericProtectedQueue is

protected body ProtectedQueue is

procedure Push (

InputItem : in Item;

Success : out Boolean) is

begin

if IsFull then

Success := False;

return;

else

Success := True;

Queue(Last) := InputItem;

Last := (Last + 1) mod Size;

Count := Count + 1;

end if;

end Push;

procedure Pop (

OutputItem : out Item;

Success : out Boolean) is

begin

if IsEmpty then

Success := False;

return;

else

Success := True;

OutputItem := Queue(First);

First := (First + 1) mod Size;

Count := Count - 1;

end if;

end Pop;

function IsFull return Boolean is

begin

return Count = Size;

end IsFull;

function IsEmpty return Boolean is

begin

return Count = 0;

end IsEmpty;

end ProtectedQueue;

end GenericProtectedQueue;

Finally, the following code defines the simulation of the Task Force Command and the Scout/Defensive Probes, allowing for a variable number of Phaser Probes to be entered and created before beginning the simulation. This code uses a queue instead of a stack.

**Boption.adb**

with RandomFunctions, AsteroidPackage, Ada.Calendar, Ada.Numerics.Float\_Random, Ada.Integer\_Text\_IO, Ada.Text\_IO, Ada.Unchecked\_Deallocation, GenericProtectedQueue;

use RandomFunctions, AsteroidPackage, Ada.Calendar,

Ada.Numerics.Float\_Random, Ada.Integer\_Text\_IO, Ada.Text\_IO;

procedure Boption is

--Declarations/Functions

procedure Free is

new Ada.Unchecked\_Deallocation(Asteroid, Asteroid\_Ptr);

package AsteroidQueuePackage is new GenericProtectedQueue(Asteroid\_Ptr,

15);

use AsteroidQueuePackage;

type DefensiveProbeType is

(Torpedos,

Phasers);

function TimeToDestroy (

Size : in Integer;

ProbeType : in DefensiveProbeType)

return Duration is

begin

if ProbeType = Torpedos then

if Size = 11 then

return Duration(6.0);

elsif Size = 9 then

return Duration(3.0);

elsif Size = 6 then

return Duration(3.0);

end if;

elsif ProbeType = Phasers then

if Size = 11 then

return Duration(6.0);

elsif Size = 9 then

return Duration(4.0);

elsif Size = 6 then

return Duration(2.0);

end if;

end if;

return Duration(0.0);

end TimeToDestroy;

--Task Types

type TFC;

type DefensiveProbeArray;

type TFC\_Ptr is access all TFC;

type DPA\_Ptr is access all DefensiveProbeArray;

task type TFC is

entry Initialize (

ID : in Integer;

TFC\_ID : in TFC\_Ptr;

DPA : in DPA\_Ptr);

entry NotifyOfAsteroid (

NewTarget : in out Asteroid\_Ptr);

entry RequestTarget (

Target : out Asteroid\_Ptr;

Available : out Boolean);

entry NotifyOfTargetDestroyed;

entry NotifyOfDeath (

DPType : in DefensiveProbeType);

end TFC;

task type ScoutProbe is

entry Initialize (

TFCParent : in TFC\_Ptr);

end ScoutProbe;

task type DefensiveProbe is

entry Initialize (

TFCParent : in TFC\_Ptr;

DPType : in DefensiveProbeType;

ProbeID : in Integer);

end DefensiveProbe;

type DPL is array (Integer range <>) of DefensiveProbe;

type DefensiveProbeArray

(NumProbes : Integer := 1) is

record

Probes : DPL (1 .. NumProbes);

end record;

--Task Bodies

task body TFC is

--ID for tracking simulations

SimID : Integer;

--Data Structures

AstStack : ProtectedQueue;

Scout : ScoutProbe;

TorpedoOne : DefensiveProbe;

TorpedoTwo : DefensiveProbe;

PhaserProbes : DPA\_Ptr;

--Mission Statistics

TorpedoDeaths : Integer := 0;

PhaserDeaths : Integer := 0;

Destroyed : Integer := 0;

Damage : Integer := 0;

MissionComplete : Boolean := False;

begin

accept Initialize (

ID : in Integer;

TFC\_ID : in TFC\_Ptr;

DPA : in DPA\_Ptr) do

SimID := ID;

PhaserProbes := DPA;

Scout.Initialize(TFC\_ID);

TorpedoOne.Initialize(TFC\_ID, Torpedos, 1);

TorpedoTwo.Initialize(TFC\_ID, Torpedos, 2);

for I in PhaserProbes.Probes'range loop

PhaserProbes.Probes(I).Initialize(TFC\_ID, Phasers, I+2);

end loop;

end Initialize;

while not MissionComplete loop

select

--Scout Probe Report New Asteroid

accept NotifyOfAsteroid (

NewTarget : in out Asteroid\_Ptr) do

--put\_line("Found new asteroid:");

--ToString(NewTarget);

declare

Success : Boolean;

begin

AstStack.Push(NewTarget, Success);

if not Success then

Damage := Damage + 1;

--put("Destroyed Asteroid #"); put(GetID(NewTarget), 2); put(" (Shields Hit)."); new\_line;

Free(NewTarget);

Destroyed := Destroyed + 1;

end if;

end;

end NotifyOfAsteroid;

or

--Defensive Probe Request Target

accept RequestTarget (

Target : out Asteroid\_Ptr;

Available : out Boolean) do

declare

TempPtr : Asteroid\_Ptr;

Success : Boolean;

begin

GetTargetLoop:

loop

AstStack.Pop(TempPtr, Success);

if Success then

if HasImpacted(TempPtr) then

--put("Destroyed Asteroid #"); put(GetID(TempPtr), 2); put(" (Shields Hit)."); new\_line;

Damage := Damage + 1;

Free(TempPtr);

Destroyed := Destroyed + 1 ;

else

Target := TempPtr;

Available := True;

exit GetTargetLoop;

end if;

else

Available := False;

exit GetTargetLoop;

end if;

end loop GetTargetLoop;

end;

end RequestTarget;

or

--Defensive Probe Report Target Destroyed

accept NotifyOfTargetDestroyed do

Destroyed := Destroyed + 1;

end NotifyOfTargetDestroyed;

or

--Defensive Probe Report Death

accept NotifyOfDeath (

DPType : in DefensiveProbeType) do

if DPType = Torpedos then

TorpedoDeaths := TorpedoDeaths + 1;

elsif DPType = Phasers then

PhaserDeaths := PhaserDeaths + 1;

end if;

Destroyed := Destroyed + 1;

end NotifyOfDeath;

end select;

--Check for mission end

if Destroyed >= 55 then

MissionComplete := True;

New\_Line;

Put("-- Mission Stats for Simulation --");

New\_Line;

Put("Status: ");

Put("SUCCESS");

New\_Line;

Put("Total # of Phaser Probes: ");

Put(PhaserProbes.Probes'Length, 2);

New\_Line;

Put("Phaser Probes Destroyed: ");

Put(PhaserDeaths, 2);

New\_Line;

Put("Total # of Torpedo Probes: 2");

New\_Line;

Put("Torpedo Probes Destroyed: ");

Put(TorpedoDeaths, 2);

New\_Line;

Put("Total # of Asteroids Destroyed: ");

Put(Destroyed, 2);

New\_Line;

Put("Shield Damage Taken: ");

Put(Damage, 2);

New\_Line(2);

elsif Damage >= 5 then

MissionComplete := True;

New\_Line;

Put("-- Mission Stats for Simulation --");

New\_Line;

Put("Status: ");

Put("FAILURE");

New\_Line;

Put("Total # of Phaser Probes: ");

Put(PhaserProbes.Probes'Length, 2);

New\_Line;

Put("Phaser Probes Destroyed: ");

Put(PhaserDeaths, 2);

New\_Line;

Put("Total # of Torpedo Probes: 2");

New\_Line;

Put("Torpedo Probes Destroyed: ");

Put(TorpedoDeaths, 2);

New\_Line;

Put("Total # of Asteroids Destroyed: ");

Put(Destroyed, 2);

New\_Line;

Put("Shield Damage Taken: ");

Put(Damage, 2);

New\_Line(2);

end if;

end loop;

end TFC;

task body ScoutProbe is

--Data

RandomNum : Generator;

Parent : TFC\_Ptr;

--Temp storage variables

Ast : Asteroid\_Ptr;

AstNumber : Integer := 1;

--Control variable

Recalled : Boolean := False;

begin

accept Initialize (

TFCParent : in TFC\_Ptr) do

Parent := TFCParent;

--Seed generator for random output

Reset(RandomNum, Integer(Seconds(Clock)));

end Initialize;

while not Recalled loop

delay Duration(RandomPoisson(4.0, RandomNum));

Ast := new Asteroid;

Construct(Ast, AstNumber, RandomSize(RandomNum), Duration(

RandomImpactTime(RandomNum)), Seconds(Clock));

AstNumber := AstNumber + 1;

Parent.NotifyOfAsteroid(Ast);

end loop;

end ScoutProbe;

task body DefensiveProbe is

--Data

Parent : TFC\_Ptr;

ProbeType : DefensiveProbeType;

ID : Integer;

--Temp storage variables

Target : Asteroid\_Ptr;

Available : Boolean;

--Control variable

SelfDestruct : Boolean := False;

begin

accept Initialize (

TFCParent : in TFC\_Ptr;

DPType : in DefensiveProbeType;

ProbeID : in Integer) do

Parent := TFCParent;

ProbeType := DPType;

ID := ProbeID;

end Initialize;

while not SelfDestruct loop

Parent.RequestTarget(Target, Available);

if Available then

if TimeToDestroy(GetSize(Target), ProbeType) <=

RemainingTimeToImpact(Target) then

--Destroy

delay TimeToDestroy(GetSize(Target), ProbeType);

--put("Destroyed Asteroid #"); put(GetID(Target), 2); put(" (D-Probe #"); put(ID, 2); put\_line(" - Destroyed)."); new\_line;

Free(Target);

Parent.NotifyOfTargetDestroyed;

if ProbeType = Torpedos then

delay 3.0;

else

delay 2.0;

end if;

else

--Self Destruct

delay RemainingTimeToImpact(Target);

--put("Destroyed Asteroid #"); put(GetID(Target), 2); put(" (D-Probe #"); put(ID, 2); put\_line(" - Self Destruct)."); new\_line;

Free(Target);

Parent.NotifyOfDeath(ProbeType);

SelfDestruct := True;

end if;

else

delay 0.5;

end if;

end loop;

end DefensiveProbe;

Simulation : aliased TFC;

Temp : DPA\_Ptr;

NumProbes : Integer;

begin

Put("Enter # of Phaser Probes: ");

Get(NumProbes);

Temp := new DefensiveProbeArray(NumProbes);

Simulation.Initialize(1, Simulation'access, Temp);

Put\_Line("Running Simulation...");

end Boption;

**PART A**

The following output was generated by running the simulation multiple times, and inputting varying numbers of Phaser Probes.

**AoptionOutput.txt**

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 1

Phaser Probes Destroyed: 1

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 31

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 2

Phaser Probes Destroyed: 2

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 28

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 3

Phaser Probes Destroyed: 3

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 35

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 4

Phaser Probes Destroyed: 4

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 34

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 5

Phaser Probes Destroyed: 5

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 29

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 6

Phaser Probes Destroyed: 6

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 40

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 7

Phaser Probes Destroyed: 7

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 37

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 8

Phaser Probes Destroyed: 8

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 47

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 9

Phaser Probes Destroyed: 9

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 4

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 10

Phaser Probes Destroyed: 8

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 1

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 11

Phaser Probes Destroyed: 11

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 4

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 12

Phaser Probes Destroyed: 10

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 2

-- Mission Stats for Simulation --

Status: FAILURE

Total # of Phaser Probes: 13

Phaser Probes Destroyed: 13

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 49

Shield Damage Taken: 5

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 14

Phaser Probes Destroyed: 8

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 1

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 0

-- Mission Stats for Simulation --

Status: SUCCESS

Total # of Phaser Probes: 15

Phaser Probes Destroyed: 10

Total # of Torpedo Probes: 2

Torpedo Probes Destroyed: 2

Total # of Asteroids Destroyed: 55

Shield Damage Taken: 2

**Simulation Statistics for Part A**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of Phaser Probes** | **Phaser Probes Destroyed** | **Torpedo Probes Destroyed** | **Asteroids Destroyed** | **Shield Damage** | **Result** |
| 1 | 1 | 2 | 31 | 5 | FAILURE |
| 2 | 2 | 2 | 28 | 5 | FAILURE |
| 3 | 3 | 2 | 35 | 5 | FAILURE |
| 4 | 4 | 2 | 34 | 5 | FAILURE |
| 5 | 5 | 2 | 29 | 5 | FAILURE |
| 6 | 6 | 2 | 40 | 5 | FAILURE |
| 7 | 7 | 2 | 37 | 5 | FAILURE |
| 8 | 8 | 2 | 47 | 5 | FAILURE |
| 9 | 9 | 2 | 55 | 4 | SUCCESS |
| 10 | 8 | 2 | 55 | 1 | SUCCESS |
| 11 | 11 | 2 | 55 | 4 | SUCCESS |
| 12 | 10 | 2 | 55 | 2 | SUCCESS |
| 13 | 13 | 2 | 49 | 5 | FAILURE |
| 14 | 8 | 1 | 55 | 0 | SUCCESS |
| 15 | 10 | 2 | 55 | 2 | SUCCESS |

**Part A Discussion**

The results for A are interesting in that they seem to suggest something different than both the C or B option. Here, the nine probe mark marks the first instance of a successful simulation, much earlier than the twelve probe mark suggested by the previous sections. However, like the eleven probe mark, the nine probe mark is barely successful, being only one shield hit away from a failed mission. Because of this, the safest bet for a minimum number of probes as suggested by these results is ten, as this marks the first successful simulation with remaining probes and minimal shield damage.  
 Probably the most interesting difference between this data and the B and C data is that the number of asteroids that can be dealt with using a smaller number of Phaser Probes is drastically higher. This is obviously the result of the priority placed on impact time by using an ordered queue; because immediate impact asteroids are prioritized, the probes are more likely to not waste time destroying asteroids that do not pose an immediate danger while the asteroids that do slam into the shields. This results in a more efficient use of resources, allowing the fleet to get away with a lot more asteroids destroyed using a lot less probes. Because this is still a queue like in the B option, one would expect the rate of asteroid destruction to scale linearly, which it does. However, comparing the results to the B option results, it becomes clear that although the A option hits the success mark much earlier, it does not scale as fast as the B option does. This seems to be the result of a higher rate of probe destruction; comparing the fifteen probe entry of the B and C option with this one, the A option result has almost twice the total number of probes destroyed. In fact, mostly all simulations show higher numbers of probes destroyed when compared to the B and C option results. Indeed, this high rate of probe destruction appears to also be the cause of the thirteen probe simulation’s failure. Considering the nature of the order queue, though, it becomes clear why the high destruction rate of probes is prevalent in this data; despite being a queue it is still possible for probes to starve. If an asteroid is inserted and its impact time is distant, it will sit in the back. If enough asteroids enter the queue with a closer impact time, then this asteroid will simply sit at the back while it gets closer to the ship, potentially resulting in a collision. This means that instead of the remaining time until impact being random when assigned to a probe like in the B option, here the remaining time approaches zero as more and more asteroids enter the system. Because of this, probes being assigned asteroids will more often than not be assigned asteroids with very short remaining impact times, increasing the chances that whatever probe is assigned the next asteroid will not be able to destroy it without destroying himself, as the remaining time will be too short.  
 Overall I would say the A option met my expectations, which was that it would be much more efficient and most likely would result in a smaller minimum number of probes. I was not expecting the A option to be so taxing on the probe count, but after careful though this makes sense. I would choose the A option as the best strategy of the three tested here, but I would not consider it the best possible strategy. There are still opportunities missed by the A option, such as taking into account the efficiency of the probes relative to the asteroid’s size. Another factor to consider is that in some cases it may be better to let an asteroid hit the shield instead of sacrificing a probe for it, because as the number of probes decrease, so too does the ability for the fleet to handle incoming asteroids; if an asteroid is going to cause the death of a probe no matter what, one could instead sacrifice the health of the shield so as to keep the probe alive.

**Part A Code**

**RandomFunctions.ads/adb** – See C option Code.

**AsteroidPackage.ads/adb** – See C option Code.

The following code defines the protected Ordered Queue used in the A option.

**GenericProtectedOrderedQueue.ads**

generic

type Item is private;

type PriorityType is private;

with function GetPriority (

Input : in Item)

return PriorityType;

with function ">=" (

A,

B : in PriorityType)

return Boolean;

Size :

in Positive;

package GenericProtectedOrderedQueue is

subtype Item\_Range is Integer range 0..(Size-1);

type Item\_Array is array (Item\_Range) of Item;

protected type ProtectedOrderedQueue is

procedure Push (

InputItem : in Item;

Success : out Boolean);

procedure Pop (

OutputItem : out Item;

Success : out Boolean);

function IsFull return Boolean;

function IsEmpty return Boolean;

private

Queue: Item\_Array;

First : Integer := 0;

Last : Integer := 0;

Count : Integer := 0;

end ProtectedOrderedQueue;

end GenericProtectedOrderedQueue;

**GenericProtectedOrderedQueue.adb**

package body GenericProtectedOrderedQueue is

protected body ProtectedOrderedQueue is

procedure Push (

InputItem : in Item;

Success : out Boolean) is

Temp1 : Integer;

Temp2 : Integer;

begin

if IsFull then

Success := False;

return;

else

Success := True;

if IsEmpty then

--Dont check just insert

Queue(Last) := InputItem;

Last := (Last + 1) mod Size;

Count := Count + 1;

else

--Find insertion spot

Temp1 := First;

while Temp1 /= Last and then GetPriority(InputItem) >=

GetPriority(Queue(Temp1)) loop

Temp1 := (Temp1 + 1) mod Size;

end loop;

if Temp1 = Last then

--Insert at end, dont need to move anything

Queue(Last) := InputItem;

Last := (Last + 1) mod Size;

Count := Count + 1;

elsif Temp1 = First then

--Insert at beginning, dont need to move anything

First := (First + Size - 1) mod Size;

Queue(First) := InputItem;

Count := Count + 1;

else

--Shift everything from Temp1 to Last over

Temp2 := Last;

while Temp2 /= Temp1 loop

Queue(Temp2) := Queue((Temp2 + Size - 1) mod Size);

Temp2 := (Temp2 + Size - 1) mod Size;

end loop;

--Insert in new open spot

Queue(Temp1) := InputItem;

Last := (Last + 1) mod Size;

Count := Count + 1;

end if;

end if;

end if;

end Push;

procedure Pop (

OutputItem : out Item;

Success : out Boolean) is

begin

if IsEmpty then

Success := False;

return;

else

Success := True;

OutputItem := Queue(First);

First := (First + 1) mod Size;

Count := Count - 1;

end if;

end Pop;

function IsFull return Boolean is

begin

return Count = Size;

end IsFull;

function IsEmpty return Boolean is

begin

return Count = 0;

end IsEmpty;

end ProtectedOrderedQueue;

end GenericProtectedOrderedQueue;

Finally, the following code defines the simulation of the Task Force Command and the Scout/Defensive Probes, allowing for a variable number of Phaser Probes to be entered and created before beginning the simulation. This code uses an ordered queue instead of a regular queue or stack.

**Aoption.adb**

with RandomFunctions, AsteroidPackage, Ada.Calendar, Ada.Numerics.Float\_Random, Ada.Integer\_Text\_IO, Ada.Text\_IO, Ada.Unchecked\_Deallocation, GenericProtectedOrderedQueue;

use RandomFunctions, AsteroidPackage, Ada.Calendar,

Ada.Numerics.Float\_Random, Ada.Integer\_Text\_IO, Ada.Text\_IO;

procedure Aoption is

--Declarations/Functions

procedure Free is

new Ada.Unchecked\_Deallocation(Asteroid, Asteroid\_Ptr);

type DefensiveProbeType is

(Torpedos,

Phasers);

function AstGTE (

A,

B : in Duration)

return Boolean is

begin

return A >= B;

end AstGTE;

function TimeToDestroy (

Size : in Integer;

ProbeType : in DefensiveProbeType)

return Duration is

begin

if ProbeType = Torpedos then

if Size = 11 then

return Duration(6.0);

elsif Size = 9 then

return Duration(3.0);

elsif Size = 6 then

return Duration(3.0);

end if;

elsif ProbeType = Phasers then

if Size = 11 then

return Duration(6.0);

elsif Size = 9 then

return Duration(4.0);

elsif Size = 6 then

return Duration(2.0);

end if;

end if;

return Duration(0.0);

end TimeToDestroy;

package AsteroidOrderedQueuePackage is new GenericProtectedOrderedQueue(

Asteroid\_Ptr, Duration, GetAbsoluteImpactTime, AstGTE, 15);

use AsteroidOrderedQueuePackage;

--Task Types

type TFC;

type DefensiveProbeArray;

type TFC\_Ptr is access all TFC;

type DPA\_Ptr is access all DefensiveProbeArray;

task type TFC is

entry Initialize (

ID : in Integer;

TFC\_ID : in TFC\_Ptr;

DPA : in DPA\_Ptr);

entry NotifyOfAsteroid (

NewTarget : in out Asteroid\_Ptr);

entry RequestTarget (

Target : out Asteroid\_Ptr;

Available : out Boolean);

entry NotifyOfTargetDestroyed;

entry NotifyOfDeath (

DPType : in DefensiveProbeType);

end TFC;

task type ScoutProbe is

entry Initialize (

TFCParent : in TFC\_Ptr);

end ScoutProbe;

task type DefensiveProbe is

entry Initialize (

TFCParent : in TFC\_Ptr;

DPType : in DefensiveProbeType;

ProbeID : in Integer);

end DefensiveProbe;

type DPL is array (Integer range <>) of DefensiveProbe;

type DefensiveProbeArray

(NumProbes : Integer := 1) is

record

Probes : DPL (1 .. NumProbes);

end record;

--Task Bodies

task body TFC is

--ID for tracking simulations

SimID : Integer;

--Data Structures

AstStack : ProtectedOrderedQueue;

Scout : ScoutProbe;

TorpedoOne : DefensiveProbe;

TorpedoTwo : DefensiveProbe;

PhaserProbes : DPA\_Ptr;

--Mission Statistics

TorpedoDeaths : Integer := 0;

PhaserDeaths : Integer := 0;

Destroyed : Integer := 0;

Damage : Integer := 0;

MissionComplete : Boolean := False;

begin

accept Initialize (

ID : in Integer;

TFC\_ID : in TFC\_Ptr;

DPA : in DPA\_Ptr) do

SimID := ID;

PhaserProbes := DPA;

Scout.Initialize(TFC\_ID);

TorpedoOne.Initialize(TFC\_ID, Torpedos, 1);

TorpedoTwo.Initialize(TFC\_ID, Torpedos, 2);

for I in PhaserProbes.Probes'range loop

PhaserProbes.Probes(I).Initialize(TFC\_ID, Phasers, I+2);

end loop;

end Initialize;

while not MissionComplete loop

select

--Scout Probe Report New Asteroid

accept NotifyOfAsteroid (

NewTarget : in out Asteroid\_Ptr) do

--put\_line("Found new asteroid:");

--ToString(NewTarget);

declare

Success : Boolean;

begin

AstStack.Push(NewTarget, Success);

if not Success then

Damage := Damage + 1;

--put("Destroyed Asteroid #"); put(GetID(NewTarget), 2); put(" (Shields Hit)."); new\_line;

Free(NewTarget);

Destroyed := Destroyed + 1;

end if;

end;

end NotifyOfAsteroid;

or

--Defensive Probe Request Target

accept RequestTarget (

Target : out Asteroid\_Ptr;

Available : out Boolean) do

declare

TempPtr : Asteroid\_Ptr;

Success : Boolean;

begin

GetTargetLoop:

loop

AstStack.Pop(TempPtr, Success);

if Success then

if HasImpacted(TempPtr) then

--put("Destroyed Asteroid #"); put(GetID(TempPtr), 2); put(" (Shields Hit)."); new\_line;

Damage := Damage + 1;

Free(TempPtr);

Destroyed := Destroyed + 1 ;

else

Target := TempPtr;

Available := True;

exit GetTargetLoop;

end if;

else

Available := False;

exit GetTargetLoop;

end if;

end loop GetTargetLoop;

end;

end RequestTarget;

or

--Defensive Probe Report Target Destroyed

accept NotifyOfTargetDestroyed do

Destroyed := Destroyed + 1;

end NotifyOfTargetDestroyed;

or

--Defensive Probe Report Death

accept NotifyOfDeath (

DPType : in DefensiveProbeType) do

if DPType = Torpedos then

TorpedoDeaths := TorpedoDeaths + 1;

elsif DPType = Phasers then

PhaserDeaths := PhaserDeaths + 1;

end if;

Destroyed := Destroyed + 1;

end NotifyOfDeath;

end select;

--Check for mission end

if Destroyed >= 55 then

MissionComplete := True;

New\_Line;

Put("-- Mission Stats for Simulation --");

New\_Line;

Put("Status: ");

Put("SUCCESS");

New\_Line;

Put("Total # of Phaser Probes: ");

Put(PhaserProbes.Probes'Length, 2);

New\_Line;

Put("Phaser Probes Destroyed: ");

Put(PhaserDeaths, 2);

New\_Line;

Put("Total # of Torpedo Probes: 2");

New\_Line;

Put("Torpedo Probes Destroyed: ");

Put(TorpedoDeaths, 2);

New\_Line;

Put("Total # of Asteroids Destroyed: ");

Put(Destroyed, 2);

New\_Line;

Put("Shield Damage Taken: ");

Put(Damage, 2);

New\_Line(2);

elsif Damage >= 5 then

MissionComplete := True;

New\_Line;

Put("-- Mission Stats for Simulation --");

New\_Line;

Put("Status: ");

Put("FAILURE");

New\_Line;

Put("Total # of Phaser Probes: ");

Put(PhaserProbes.Probes'Length, 2);

New\_Line;

Put("Phaser Probes Destroyed: ");

Put(PhaserDeaths, 2);

New\_Line;

Put("Total # of Torpedo Probes: 2");

New\_Line;

Put("Torpedo Probes Destroyed: ");

Put(TorpedoDeaths, 2);

New\_Line;

Put("Total # of Asteroids Destroyed: ");

Put(Destroyed, 2);

New\_Line;

Put("Shield Damage Taken: ");

Put(Damage, 2);

New\_Line(2);

end if;

end loop;

end TFC;

task body ScoutProbe is

--Data

RandomNum : Generator;

Parent : TFC\_Ptr;

--Temp storage variables

Ast : Asteroid\_Ptr;

AstNumber : Integer := 1;

--Control variable

Recalled : Boolean := False;

begin

accept Initialize (

TFCParent : in TFC\_Ptr) do

Parent := TFCParent;

--Seed generator for random output

Reset(RandomNum, Integer(Seconds(Clock)));

end Initialize;

while not Recalled loop

delay Duration(RandomPoisson(4.0, RandomNum));

Ast := new Asteroid;

Construct(Ast, AstNumber, RandomSize(RandomNum), Duration(

RandomImpactTime(RandomNum)), Seconds(Clock));

AstNumber := AstNumber + 1;

Parent.NotifyOfAsteroid(Ast);

end loop;

end ScoutProbe;

task body DefensiveProbe is

--Data

Parent : TFC\_Ptr;

ProbeType : DefensiveProbeType;

ID : Integer;

--Temp storage variables

Target : Asteroid\_Ptr;

Available : Boolean;

--Control variable

SelfDestruct : Boolean := False;

begin

accept Initialize (

TFCParent : in TFC\_Ptr;

DPType : in DefensiveProbeType;

ProbeID : in Integer) do

Parent := TFCParent;

ProbeType := DPType;

ID := ProbeID;

end Initialize;

while not SelfDestruct loop

Parent.RequestTarget(Target, Available);

if Available then

if TimeToDestroy(GetSize(Target), ProbeType) <=

RemainingTimeToImpact(Target) then

--Destroy

delay TimeToDestroy(GetSize(Target), ProbeType);

--put("Destroyed Asteroid #"); put(GetID(Target), 2); put(" (D-Probe #"); put(ID, 2); put\_line(" - Destroyed)."); new\_line;

Free(Target);

Parent.NotifyOfTargetDestroyed;

if ProbeType = Torpedos then

delay 3.0;

else

delay 2.0;

end if;

else

--Self Destruct

delay RemainingTimeToImpact(Target);

--put("Destroyed Asteroid #"); put(GetID(Target), 2); put(" (D-Probe #"); put(ID, 2); put\_line(" - Self Destruct)."); new\_line;

Free(Target);

Parent.NotifyOfDeath(ProbeType);

SelfDestruct := True;

end if;

else

delay 0.5;

end if;

end loop;

end DefensiveProbe;

Simulation : aliased TFC;

Temp : DPA\_Ptr;

NumProbes : Integer;

begin

Put("Enter # of Phaser Probes: ");

Get(NumProbes);

Temp := new DefensiveProbeArray(NumProbes);

Simulation.Initialize(1, Simulation'access, Temp);

Put\_Line("Running Simulation...");

end Aoption;