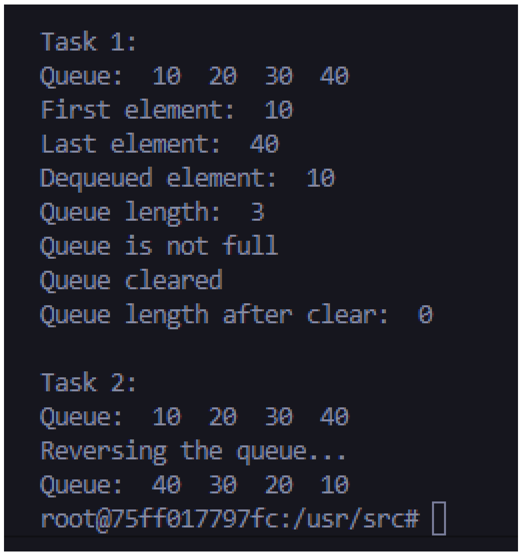
Name: Luka Aitken

Student ID: T00663672

Date: March 3, 2025

SENG 4630 – Lab 7 – Private Packages and Types

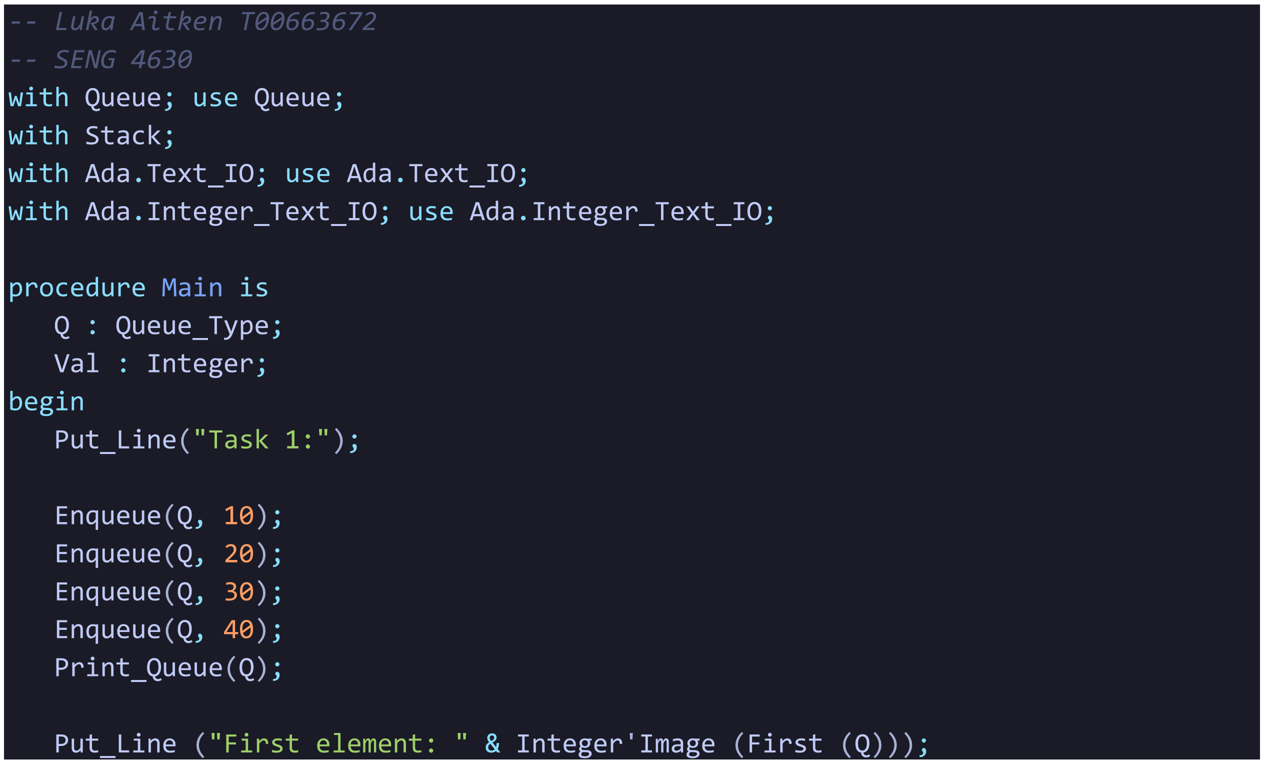
Outputs for Exercise 1 & 2:



```
Task 1:
Queue:  10  20  30  40
First element:  10
Last element:  40
Dequeued element:  10
Queue length:  3
Queue is not full
Queue cleared
Queue length after clear:  0

Task 2:
Queue:  10  20  30  40
Reversing the queue...
Queue:  40  30  20  10
root@75ff017797fc:/usr/src#
```

main.adb:



```ada
-- Luka Aitken T00663672
-- SENG 4630
with Queue; use Queue;
with Stack;
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Main is
   Q : Queue_Type;
   Val : Integer;
begin
   Put_Line("Task 1:");

   Enqueue(Q, 10);
   Enqueue(Q, 20);
   Enqueue(Q, 30);
   Enqueue(Q, 40);
   Print_Queue(Q);

   Put_Line ("First element: " & Integer'Image (First (Q)));
   Put_Line ("Last element: " & Integer'Image (Last (Q)));

   Dequeue (Q, Val);
   Put_Line ("Dequeued element: " & Integer'Image (Val));

   Put_Line ("Queue length: " & Integer'Image (Length (Q)));

   if Is_Full (Q) then
      Put_Line ("Queue is full");
   else
      Put_Line ("Queue is not full");
   end if;

   Clear (Q);
   Put_Line ("Queue cleared");
   Put_Line ("Queue length after clear: " & Integer'Image (Length (Q)));


   Put_Line("");
   Put_Line("Task 2:");
   Enqueue(Q, 10);
   Enqueue(Q, 20);
   Enqueue(Q, 30);
   Enqueue(Q, 40);
   Print_Queue(Q);
   Put_Line("Reversing the queue...");
   reversequeue(Q);
   Print_Queue(Q);
end Main;
```

queue.adb:

```ada
-- Luka Aitken T00663672
-- SENG 4630
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Stack; use Stack;

package body Queue is

   procedure Enqueue(Q : in out Queue_Type; Val : Integer) is
   begin
      if Q.Count = Queue_Index'Last then
         Put_Line("Error: Queue is full.");
      else
         Q.Rear := (Q.Rear mod Queue_Index'Last) + 1;
         Q.Data(Q.Rear) := Val;
         Q.Count := Q.Count + 1;
      end if;
   end Enqueue;

   procedure Dequeue(Q : in out Queue_Type; Val : out Integer) is
   begin
      if Q.Count = 0 then
         Put_Line("Error: Queue is empty.");
         Val := 0;
      else
         Val := Q.Data(Q.Front);
         Q.Front := (Q.Front mod Queue_Index'Last) + 1;
         Q.Count := Q.Count - 1;
      end if;
   end Dequeue;

   function First(Q : Queue_Type) return Integer is
   begin
      if Q.Count = 0 then
         Put_Line("Queue is empty.");
         return 0;
      end if;
      return Q.Data(Q.Front);
   end First;

   function Last(Q : Queue_Type) return Integer is
   begin
      if Q.Count = 0 then
         Put_Line("Queue is empty.");
         return 0;
      end if;
      return Q.Data(Q.Rear);
   end Last;

   function Length(Q : Queue_Type) return Integer is
   begin
      return Q.Count;
   end Length;

   function Is_Full(Q : Queue_Type) return Boolean is
   begin
      return Q.Count = Queue_Index'Last;
   end Is_Full;
```

```
   procedure Clear(Q : in out Queue_Type) is
   begin
      Q.Front := 1;
      Q.Rear := 0;
      Q.Count := 0;
   end Clear;

   procedure Print_Queue(Q : Queue_Type) is
      I : Natural := Q.Front;
      Count : Integer := Q.Count;
   begin
      if Count = 0 then
         Put_Line("Queue is empty.");
         return;
      end if;

      Put("Queue: ");
      for J in 1 .. Count loop
         Put(Integer'Image(Q.Data(I)) & " ");
         I := (I mod Queue_Index'Last) + 1;
      end loop;
      New_Line;
   end Print_Queue;

   procedure reversequeue(Q : in out Queue_Type) is
      Stack1 : Stack.Stack_Type;
      Temp   : Integer;
   begin
      while Q.Count > 0 loop
         Dequeue(Q, Temp);
         Push(Stack1, Temp);
      end loop;

      while not Is_Empty(Stack1) loop
         Pop(Stack1, Temp);
         Enqueue(Q, Temp);
      end loop;
   end reversequeue;

end Queue;
```

queue.ads:

```ada
-- Luka Aitken T00663672
-- SENG 4630
with Stack; use Stack;
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

package Queue is
   subtype Queue_Index is Natural range 1 .. 10;
   type Queue_Type is private;

   procedure Enqueue(Q : in out Queue_Type; Val : Integer);
   procedure Dequeue(Q : in out Queue_Type; Val : out Integer);
   function First(Q : Queue_Type) return Integer;
   function Last(Q : Queue_Type) return Integer;
   function Length(Q : Queue_Type) return Integer;
   function Is_Full(Q : Queue_Type) return Boolean;
   procedure Clear(Q : in out Queue_Type);
   procedure Print_Queue(Q : Queue_Type);
   procedure reversequeue(Q: in out Queue_Type);

private
   type Content_Type is array(Queue_Index) of Integer;
   type Queue_Type is record
      Data  : Content_Type;
      Front : Natural := 1;
      Rear  : Natural := 0;
      Count : Natural := 0;
   end record;
end Queue;
```

Stack.adb:

```ada
-- Luka Aitken T00663672
-- SENG 4630
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Stack;

package body Stack is

   procedure Push(S : in out Stack_Type; Val : Integer) is
   begin
      if S.Top = 10 then
         Put_Line("Error: Stack is full.");
      else
         S.Top := S.Top + 1;
         S.Data(S.Top) := Val;
      end if;
   end Push;

   procedure Pop(S : in out Stack_Type; Val : out Integer) is
   begin
      if S.Top = 0 then
         Put_Line("Error: Stack is empty.");
         Val := 0;
      else
         Val := S.Data(S.Top);
         S.Top := S.Top - 1;
      end if;
   end Pop;

   function Is_Empty(S : Stack_Type) return Boolean is
   begin
      return S.Top = 0;
   end Is_Empty;

end Stack;
```

Stack.ads:

```
-- Luka Aitken T00663672
-- SENG 4630
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

package Stack is
   type Stack_Type is private;

   procedure Push(S : in out Stack_Type; Val : Integer);
   procedure Pop(S : in out Stack_Type; Val : out Integer);
   function Is_Empty(S : Stack_Type) return Boolean;

private
   type Content_Type is array (1 .. 10) of Integer;
   type Stack_Type is record
      Top    : Natural := 0;
      Data   : Content_Type;
```

```ada
   end record;
end Stack;
```