University of Puerto Rico
Mayagüez Campus
College of Engineering
Department of Electrical and Computer Engineering

ICOM4029 – Compilers
Professor: Bienvenido Vélez
Technical Assistant: René D. Badía

Laboratory 4 – PA's 1 & 2

I. Solution to Programming Assignment 1

Here's a sample solution to PA1:

```
class Stack inherits IO{
      (*
         Class Stack uses roughly the same code as class List
         in the samples except that the elements are Strings and
         some stack operations are added.
         It defines empty stacks.
      *)

      --isNil is used to check if the stack is empty
      isNil() : Bool {true};

      --head returns the top element, which does not exist on empty
      --stacks so it is an error.
      top()  : String { { abort(); ""; } };

      --tail returns a Stack containing elements after the top element
      --This is an error on empty stacks.
      rest()  : Stack { { abort(); self; } };

      --push returns a new stack with argument "i" pushed.
      --If something is pushed, now the stack is not empty and is of
      --dynamic type NonEmptyStack.
      --   i = the String to push into the stack
      push(i : String) : Stack {
        (new NonEmptyStack).init(i, self)
      };

      --pop returns a new stack without the original top element.
      pop() : Stack{ rest() };

      --print displays the contents of the stack.
      --It prints the top of the stack and then recursively prints
      --the rest.
      --   l = the stack to be printed
      print(l : Stack) : Object {
            if l.isNil() then 0
            else
            {
                  out_string(l.top().concat("\n"));
                  print(l.rest());
            }
            fi
      };
};

class NonEmptyStack inherits Stack {
      -- Class NonEmptyStack defines non-empty stacks.
      top : String; --the top element
      rest : Stack; --a stack containing the rest of the elements

      isNil() : Bool { false };
      top()   : String { top };
      rest()  : Stack { rest };

      -- Initializes a NonEmptyStack object
      --    i = the initial top element
      --    s = the initial stack with the rest of the elements
      init(i : String, s : Stack) : Stack {
      {
            top <- i;
            rest <- s;
            self;
      }
      };
};

class Main inherits IO {
      theStack : Stack <- new Stack; --the stack object
      inStr : String;                --for storing the command
      x : Bool <- false;             --for exiting the program loop
      temp1 : String;                --for storing an element
      temp2 : String;                --for storing a 2nd element

      --switch switches the top 2 elements of the stack
      switch() : Object {
      {
            -- temporarily store and pop the next 2 elements
            temp1 <- theStack.top();
            theStack <- theStack.pop();
            temp2 <- theStack.top();
            theStack <- theStack.pop();

            -- push them in reverse order
            theStack <- theStack.push(temp1);
            theStack <- theStack.push(temp2);
      }
      };

      --sum adds the top 2 elements of the stack and pushes the result
      sum() : Object {
            let tempNum : Int, conv : A2I <- new A2I in
            {
                  -- temporarily store and pop the the two operands
                  temp1 <- theStack.top();
                  theStack <- theStack.pop();
                  temp2 <- theStack.top();
                  theStack <- theStack.pop();

                  -- calculate the sum
                  tempNum <- conv.a2i(temp1) + conv.a2i(temp2);
                  -- push the result
                  theStack <- theStack.push(conv.i2a(tempNum));
            }
      };

      --eval does the following:
      -- *if the top is 's' then switch the following 2 elements
      -- *if it is '+', add the 2 following elements and store the sum
      -- *otherwise, do nothing
      eval() : Object {
            if theStack.isNil() then 0
            else
            {
                  -- get the top of the stack
                  temp1 <- theStack.top();
                  if temp1 = "s" then
                  {
                        theStack <- theStack.pop();
                        --switch the following 2 elements
                        switch();
                  }
                  else if temp1 = "+" then
                  {
                        --sum the following 2 elements
                        theStack <- theStack.pop();
                        sum();
                  }
                  else 0
                  fi fi;
            }
            fi
      };

      --prompt asks for a string to be entered and returns it
      prompt() : String {
      {
            out_string(">");
            in_string();
      }
      };

      --main program function
      main() : Object {
            let x : Bool <- false in
            {
                  out_string("\nStack Machine Initialized\n\n");
                  --loop until 'x' is entered
                  while (x = false) loop
                  {
                        --prompt for the command
                        inStr <- prompt();
                        if inStr = "x" then
                        {
                              --force exit from loop
                              x <- true;
                        }
```

```
                        else if inStr = "d" then
                        {
                              --print the stack
                              theStack.print(theStack);
                        }
                        else if inStr = "e" then
                        {
                              --evaluate the top of the stack
                              eval();
                        }
                        else
                        {
                              --push the input into the stack
                              theStack <- theStack.push(inStr);
                        }
                        fi fi fi;
                  }
                  pool;
                  -- exit
                  out_string("Bye!\n");
            }
      };
};
```

II. Questions about PA2