

Concepts of Programming Languages (INFOTECH)

Summer Semester 2007
Prof. Plödereder, Stefan Staiger

Assignment #2

Please submit your solution on Friday, June 29th 2007 in class or send it via email to staiger@informatik.uni-stuttgart.de. (If you want to submit earlier, please do so.) Please submit either plain text or PDF, but not MS Word files.

1 Parameter Passing Mechanisms

The following source code is a program written in *pseudo-Ada* (i.e., one should not assume that the program uses the Ada parameter passing mechanism).

```
type Field is array (1..5) of Integer;
```

```
a : Field;
```

```
procedure p (x,y : Field; i : Integer) is
begin
  x (i) := y (6-i);
  if i < 5 then
    p (x,y, i+1);
  end if;
end p;
```

```
procedure main is
begin
  a := (1, 2, 3, 4, 5);

  p (a,a,1);
end main;
```

- a. What is the value of variable **a** after the call `p(a,a)` assuming the following parameter passing mechanisms for `x` and `y`? (assume that `i` is always passed by value)

- Call by value
- Call by reference
- Call by value/result (a.k.a. copy/restore)
- Call by name

Draw the activation records on the stack to illustrate step-by-step what happens!

- b. Explain the meaning of *aliasing* (by using a small example). Which parameter passing mechanisms cause aliasing in the above example program?
- c. Given a choice for each parameter, which parameter passing mechanism should you use to get the expected behavior of the program?

2 Passing arrays to procedures

The following procedure operates on 2-dimensional arrays of arbitrary size:

```
type Matrix is array (Integer range <>, Integer range <>) of Real;
type Matrix_Access is access all Matrix;
```

```
procedure Do_Something (M : Matrix) is
  I, J : Integer;
  R    : Real;
begin
  ...
  R := M (I, J);
  ...
end Do_Something;
```

Somewhere else it is called:

```
procedure Main is
  M1 : Matrix (1 .. 10, 5 .. 17);
  M2 : Matrix_Access := new Matrix (3 .. 6, 1 .. 2);
begin
  Do_Something (M1);
  Do_Something (M2.all);
end Main;
```

- Where are the contents of `M1` stored in procedure `Main`? On the stack or on the heap? Where are the contents of `M2` stored?
- For the access to `M` in the procedure `Do_Something` the array `dope` is needed. Why?
- Show the contents of the `dope`. Show how the elements of `M2.all` are arranged in memory, assuming row-major array layout.
- How is the `dope` passed to the procedure `Do_Something`?

3 Heap Management

The following part of a program allocates memory on the heap:

```
procedure Free is new Ada.Unchecked_Deallocation (...);

...

for I in 1..100000 loop
  P := new T1;
  Q := new T2;
  ...
  Ada.Text_IO.Put_Line ('...');
  ...
  Free (P);
  Free (Q);
end loop;
```

Assume that the type T1 needs 20 bytes of storage and the type T2 needs 32 bytes of storage. Each pointer needs 4 bytes of storage. No allocations happen before the loop starts.

The heap management uses a freelist. Allocation always searches the freelist from the beginning until a block of memory is found that is large enough. When the block is larger than required, the block is split into two parts: the first is allocated and the second is put into the freelist again.

Deallocation of a block of memory prepends the block to the freelist. It doesn't merge any blocks in the freelist.

- a. How many blocks of memory does the freelist contain after the loop is finished?
How many bytes of memory are then contained in the small blocks in the freelist?
- b. What effects could the user notice when the program is run?
- c. How can the problem be corrected?
- d. Does the bitvector heap management scheme have this problem, too?