A

Student Number: __________

Family Name: __________

Given Names: __________

Signature: __________

THE UNIVERSITY OF NEW SOUTH WALES

Midsession Exam

30 August 2001

COMP 1011

Computing 1A

- Time allowed: **60 minutes**
- Total number of questions: **21**
- Each question has five alternatives of which exactly **one** is correct. There are 21 questions. You must answer Question 1, for which there is no mark. If you do not answer Question 1 properly, your exam might not be marked correctly and you risk a mark of 0. Each of Question 2–21 is worth 1 mark. For each question, you will receive 0 marks for no answer, a wrong answer, or multiple answers.
- The questions **are** of equal value
- Answer questions on the answer sheet **only**; answers on this paper are disregarded
- This paper may **not** be retained by the candidate
- Write with pencil **only**
- Closed book examination; mobile phones must be switched **off** and placed **under** the chair
- There is a 5% penalty if you do not follow the examination instructions; in particular if you do not fill in your student number and name correctly.

Question 1

Which letter appears in the upper left corner of the frontpage of this exam paper?

(A) The letter “A”

(B) The letter “B”

(C) The letter “C”

(D) The letter “D”

(E) The letter “E”

Question 2

Which of the following is the UNIX command to change file permissions?

(A) `chmod`

(B) `ls -l`

(C) `pwd`

(D) `ghc`

(E) None of the above

Question 3

Which of the following UNIX commands will search for all lines containing the word `Haskell` in the file `important.txt` and output the result into a file `result.txt`?

(A) `grep Haskell important.txt >result.txt`

(B) `grep Haskell important.txt result.txt`

(C) `grep Haskell important.txt | grep result.txt`

(D) `grep Haskell important.txt | result.txt`

(E) None of the above

Question 4

Which of the following statements regarding file permissions in UNIX is wrong? (If A to D are correct, select E.)

(A) The superuser of a UNIX system can access all files

(B) Each user can change the permission owned by that users

(C) Each user can access the files of everybody else

(D) UNIX distinguishes between the permissions for the owner of a file and other users

(E) None of the above; i.e., all statements are correct

Question 5

Which pattern does **not** match the list `[[1], [2], []]`?

(A) `x1 : (x2 : x3)`

(B) `(xs : x)`

(C) `(x : [])`

(D) `(x1 : x2) : xs`

(E) All of the above pattern match

Question 6

Which of the following expressions will generate an error?

(A) `3:[4, 5]`

(B) `[5, "Hello"]`

(C) `[1,2] ++ (1:[])`

(D) Two of the above

(E) None of the above

Question 7

Consider the following *incomplete* Haskell function that is supposed to add all elements of a list:

```
addList        :: [Int] -> Int
addList []      = 0
addList (x:xs)  = ⟨missing piece⟩
```

Which of the following pieces of code correctly completes the above function?

(A) `addList xs`

(B) `x + addList xs`

(C) `x + xs`

(D) `addList x xs`

(E) None of the above

Question 8

Provided a function of type

```
foo :: [a] -> [b] -> [(a, b)]
```

which of the following applications of `foo` result in a type error?

(A) `foo [] []`

(B) `foo [1, 2] ["Hello", "World"]`

(C) `foo [1, 2] [3, 4]`

(D) `foo [1, 2] [True]`

(E) None of the above; i.e., all applications are type correct

Question 9

Which of the following statements is correct?

(A) A partial function is a function without a type signature
(B) A partial function is not defined for some input values
(C) A partial function is a recursive function without a base case
(D) A partial function cannot be compiled
(E) None of the above statements is correct

Question 10

Both lists and tuples can be used to form collections of basic values. Which of the statements A to D is wrong? (If A to D are correct, select E.)

(A) Lists require all elements to be of the same type
(B) Tuples require that the number of arguments is fixed
(C) Tuples require all arguments to be of the same type
(D) Lists can be of varying length
(E) All of the above statements are correct

Question 11

Which of the following statements about the expression `[] : []` is true?

(A) The expression contains a type error
(B) The expression is invalid and should be replaced by `[] ++ []`
(C) The expression is valid and evaluates to the empty list
(D) The expression is valid and evaluates to a list with a single element
(E) None of the above

Question 12

If the variable `x` represents an integer value, how can we print it to the console in a *compiled* program?

(A) Just by returning it, i.e., just writing `x`
(B) With `return x`
(C) With `print x`
(D) With `show x`
(E) None of the above

Question 13

Which of the following statements about the eval-print loop of GHCi is correct?

(A) On entering an expression, the eval-print loop implicitly prints the result of this expression
(B) The eval-print loop is only used by a compiler
(C) The eval-print loop is implemented by `Main.main`
(D) The eval-print loop exactly behaves like a compiler
(E) None of the above statements is correct

Question 14

Consider the definition of `echoLine`, which should read a line of text from the keyboard and echo it to the screen.

```
echoLine :: IO ()
echoLine  = do
              putStr getLine
```

There is an error in this definition. How can it be corrected?

(A) Replace `IO ()` by `IO String`

(B) Replace `getLine` by `readLn`

(C) Replace `putStr getLine` by the two lines

```
input <- getLine
putStr input
```

(D) Replace `putStr getLine` by the two lines

```
input = getLine
putStr input
```

(E) None of the above corrections works

Question 15

Which of the following statements about the definition is correct?

```
baz str | str == "" = []
        | otherwise = str : baz (tail str)
```

(A) `baz "abc"` evaluates to `[]`

(B) The evaluation of `baz "abc"` does not stop

(C) `baz "abc"` evaluates to `["c","bc","abc"]`

(D) `baz "abc"` evaluates to `["abc","bc","c"]`

(E) None of the above statements is correct

Question 16

Consider the following function definition:

```
len (x:xs) = 1 + len xs
len []     = 1
```

Which of the following statements is correct?

(A) The definition, as it is, correctly returns the number of elements in a list, for any input list

(B) After swapping the first and second line of the definition, it correctly returns the number of elements in a list, for any input list

(C) The definition, as it is, correctly returns the number of elements in a list, for any input list **except** the empty list

(D) The definition, as it is, returns the number of elements in a list plus one, for any input list

(E) None of the above statements is correct

Question 17

Consider the following function definition:

```
foo []      = []
foo (x:xs)  = (foo xs) ++ [x]
```

The result of evaluating `foo [1, 2]` is

(A) `[[1], [2]]`

(B) `[[2], [1]]`

(C) `[2, 1]`

(D) `[1, 2]`

(E) None of the above

Question 18

Given the definition

```
bar []     = 0
bar (x:xs) = x + bar xs
```

Which of the following expressions leads to an error message:

(A) `bar []`

(B) `bar (3, 2, 1)`

(C) `bar [3, 2, 1]`

(D) All of the above

(E) None of the above

Question 19

The function `unlines` converts a list of strings into a single string, where each input string is followed by a newline character (forcing a line break). The following definition of `unlines` contains an error:

```
unlines        :: [String] -> String
unlines []      = []
unlines (x:xs)  = [x] ++ "\n" ++ (unlines xs)
```

Which modification will correct the error?

(A) Replace `[x]` by `x`

(B) Replace `unlines [] = []` by `unlines [] = ""`

(C) Replace `[x]` by `putStr x`

(D) Replace `[x] ++ "\n"` by `[x] : "\n"`

(E) None of the above

Question 20

Consider the following four definitions for computing the conjunction of two boolean values (i.e., logical "and" as implemented by Haskell's predefined function `&&`):

```
and1 True True = True
and1 x    y    = False

and2 True  x = x
and2 False x = False

and3 x y = if x then y else False

and4 True  True = True
and4 False False = False
```

Which of the following statements is correct?

(A) The function `and1` contains an error—i.e., it does not behave like `&&`

(B) The function `and2` contains an error—i.e., it does not behave like `&&`

(C) The function `and3` contains an error—i.e., it does not behave like `&&`

(D) The function `and4` contains an error—i.e., it does not behave like `&&`

(E) All four function definitions are correct—i.e., they all behave like `&&`

Question 21

Consider the following definitions:

```
mySqrt n = mySqrt' n 0

mySqrt' n m | n < 0     = -1
            | otherwise = 1 + mySqrt' (n - 2 * m - 1) (m + 1)
```

What is the result of evaluating `mySqrt 7`?

(A) Evaluating the expression `mySqrt 7` causes GHCi to loop forever

(B) The returned value is `2`

(C) The returned value is `3`

(D) The returned value is `-1`

(E) The returned value is greater than `2` and less than `3`