

CSE 1325-001

Practice Final Exam

Practice A

ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Multiple Choices (50 points total / 2 points each)

1. What is HyperThreading?
2. Self contained execution environment including its own memory space
3. A chip with multiple cores, ALU/Register sets runs on separate thread
4. std::thread
5. ALU with 2 register sets, interleaving 2 threads of execution
6. What is a Multi-core professor?
7. Utilize many cores
8. A chip with multiple cores, ALU/Register sets runs on separate
9. An algorithm can be paused while executing and safely executed by other thread
10. Independent path of execution with a process running concurrently with other threads within shared memory space
11. What is a process?
12. Self contained execution environment including its own memory space
13. Computer technology doubles every 2 years
14. When threads switches between microprocessor instruction
15. An algorithm can be paused while executing and safely executed by other thread
16. Which of the following is Moore’s Law?
17. Self contained execution environment including its own memory space
18. ALU with 2 register sets, interleaving 2 threads of execution
19. Computer technology doubles every 2 years
20. Independent path of execution with a process running concurrently with other threads within shared memory space
21. Which of the following is NOT concurrency?
22. Utilize many cores
23. Perform 2+ algorithms simultaneously
24. Self contained execution environment including its own memory space
25. ALU with 2 register sets, interleaving 2 threads of execution
26. Both C and D
27. Which of the following are not good uses for Concurrency?
28. Perform by processing Independent of User Unit
29. Programming logically independent program units
30. Process small individual units of a large problem
31. Perform 2+ algorithms simultaneously
32. None of the above
33. What do you call to get thread ID?
34. get\_id()
35. get\_thread\_id()
36. thread\_id()
37. id()
38. None of the above
39. What is the cause of garble?
40. Independent path of execution with a process running concurrently with other threads within shared memory space
41. When threads switches between microprocessor instruction
42. Self contained execution environment including its own memory space
43. An algorithm can be paused while executing and safely executed by other thread
44. What does Mutex do?
45. Perform by processing Independent of User Unit
46. Utilize many cores
47. Forces the parameter to pass as a reference, rather than a value
48. Solves immutable classes, marking all field as const
49. Both A and C
50. Code to check integer
51. regex integer{"-?\\d+"}
52. if(regex\_match(input,integer))
53. regex integer{"-?\\w+"}
54. if(regex\_match(input,char))
55. None of the above
56. Translate to Regex: 43+28
57. \d+\+\d+
58. \w+\+\w+
59. \d+
60. \w+
61. Either A or C
62. Regex code to check if it matched anything inside the bracket
63. [ ]
64. [^]
65. [ | ]
66. [&]
67. None of the above

Free Response (50 points total)

1. Draw an Activity UML that will return if a student passed the class or not based-on their final grade based-on the given program in C++ below.

#include <iostream>

using namespace std;

int main()

{

double grade;

cin >> grade;

if(grade < 70)

{

cout << “Student failed the class.” << endl;

}

else

{

cout << “Student passed the class.” << endl;

}

return 0;

}

1. Draw the GUI based-on the given code. Feel free to draw the GUI of the menu separately on the side.

Gtk::Dialog \*dialog = new Gtk::Dialog;

Gtk::Box \*vbox = Gtk::manage(new Gtk::Box(Gtk::ORIENTATION\_VERTICAL, 0));

dialog->add(\*vbox);

Gtk::MenuBar \*menubar = Gtk::manage(new Gtk::MenuBar());

vbox->pack\_start(\*menubar, Gtk::PACK\_SHRINK, 0);

Gtk::MenuItem \*menuitem\_file = Gtk::manage(new Gtk::MenuItem(“\_File”, true));

menubar->append(\*menuitem\_file);

Gtk::Menu \*filemenu = Gtk::manage(new Gtk::Menu());

menuitem\_file->set\_submenu(\*filemenu);

Gtk::MenuItem \*menuitem\_exit = Gtk::manage(new Gtk::MenuItem("\_Exit", true));

menuitem\_exit->signal\_activate().connect(sigc::mem\_fun(\*this, &Dialog::on\_exit\_click));

filemenu->append(\*menuitem\_exit);

Gtk::Box \*hbox1 = Gtk::manage(new Gtk::Box(Gtk::ORIENTATION\_HORIZONTAL, 0));

vbox->add(\*hbox1);

Gtk::Grid \*grid1 = Gtk::manage(new Gtk::Grid);

hbox1->add(\*grid1);

Gtk::Button \*hello\_button = Gtk::manage(new Gtk::Button(“Hello”));

grid1->attach(\*hello\_button, 0, 0, 1, 1);

Gtk::Button \*bye\_button = Gtk::manage(new Gtk::Button(“Bye”));

grid1->attach(\*bye\_button, 1, 0, 1, 1);

Gtk::Box \*hbox2 = new Gtk::manage(Gtk::Box(Gtk::ORIENTATION\_HORIZONTAL, 0));

vbox->add(\*hbox2);

Gtk::Button \*exit\_button = Gtk::manage(new Gtk::Button(“Exit”));

hbox2->pack\_start(\*exit\_button);