CS601: Software Development for Scientific Computing

Maximum Points: 25 Mid-semester examination 20/09/2022, 2:30PM to 4:30PM

Instructions: This exam has two parts. Part I is open book, open notes (printed/written). No electronic devices allowed. Part II is take-home. The submission instructions for part II are the same as in programming assignment 1. State your assumptions (if any) clearly.

Part I **(20 points)**:

1. Computing $\sqrt{x^2 + y^2}$ is a common problem. A common implementation strategy is as follows:

```
double ComputeHypotenuse(double x, double y) {
      return sqrt(x*x + y*y);
}
```

However, the above strategy is not robust. The computation can result in *overflow* when the values of x and y are large. Write a modified code snippet that overcomes this problem. Hint: If x is larger than y, note that $\sqrt{x^2 + y^2} = |x| \sqrt{1 + \left(\frac{y}{x}\right)^2}$. $|x|$ denotes absolute value of x. **(4 points)**

2. You visit a vegetable shop and observe that green leafy vegetables are kept close to each other in one section and separated from other vegetables. Name the Computer Science concept that resonates with a possible reasoning behind the leafy vegetable arrangement. You must explain your answer. **(1 point)**

3. i) A `Makefile` is shown below. When you run the `make` command, 3 commands are executed. Write the commands in the exact order in which they are executed (you may just write the line number containing the command. Assume that the programs have no syntax error or warnings). **(1.5 points)**

```
1  CXX=g++
2  CFLAGS= -g
3  OBJ= matvec_colmajor.o matvec_rowmajor.o
4
5  all: matvec_colmajor matvec_rowmajor
6          $(CXX) $(CFLAGS) $(OBJ) testmatvec.cpp -o test -lm
7
8  matvec_colmajor: matvec_colmajor.o
9
10
11 matvec_colmajor.o: matvec_colmajor.cpp
12         $(CXX) $(CFLAGS) -c $^
13
14 matvec_rowmajor: matvec_rowmajor.o
15
16
17 matvec_rowmajor.o: matvec_rowmajor.cpp
18         $(CXX) $(CFLAGS) -c $^
19 clean:
20         rm -f *.o matvec_colmajor matvec_rowmajor
```

ii) After executing `make` command, you edit `matvec_rowmajor.cpp` and again execute the `make` command. What are all the commands that are executed? Write line numbers only. **(1 point)**

iii) You change $^ at lines 12 and 18 to $<. Will there be any change? Why or why not? **(1 point)**

4.

```
1  #include<iostream>
2  class B {
3  public:
4          B( ) { }
5          ~B() { }
6          void print() {
7                  cout << "B::print( ) ";
8          }
9          void print(int i) {
10                 cout << "B::print(int)" << endl;
11         }
12 };
13
14 class D : public B {
15 public:
16         D( ) { }
17         ~D() { }
18         void print() {
19                 cout << "D::print( ) ";
20         }
21         void print(int i) {
22                 cout << "D::print(int)" << endl;
23         }
24 };
25
26 int main(int argc, char * argv[]) {
27         D *d = new D( );
28         B *b = (B*) d;;
29         b->print( );
30         d->print(4);
31         return 0;
32 }
33
```

i) What is printed? **(1 point)**
a) D::print( ) D::print(int)
b) B::print( ) D::print(int)
c) D::print( ) B::print(int)
d) B::print( ) B::print(int)
e) Compiler error because it is ambiguous which B and/or D is to be called.

(Note: the double semicolon on line 28 is not a problem)

ii) if the programmer intended to use b polymorphically in `main` on line 29 i.e. expected D's print method (on line 18) to be invoked, what needs to change? Describe all the code changes. **(0.5 points)**

5. Consider Gaxpy (y=y+Ax) and outer-product ($A=A+yx^T$) pseudocode shown below. (**4 points + 6 points**)

gaxpy (a_i is i^{th} column of matrix A)

```
1. for i=1 to n
2.   for j=1 to n
3.     y[j]=y[j]+a_i x[j]
```

outer-product

```
1. for j=1 to n
2.   for i=1 to n
3.     a_ij= a_ij +y[i]*x[j]
```

i) What is the exact arithmetic cost involved in each case? assume each of addition and multiplication as a step.
ii) What is the exact data movement overhead involved in each case? Assume that a vector of n elements can be read/written with one memory read/write operation. You must explain your answer with the help of line numbers shown in the pseudo code. No points awarded without explanation.

Part II **(5 points)** *You must answer questions 1 and 2 OR question 3. If an undergraduate student team answers question 3, you will be considered for bonus points. If a team with at least 1 PG student answers question 3, you will be considered for PG course credits.* Tag all your changes made to the PA1 repo as `cs601midsemsubmission`

1) Add a rule to your PA1 `makefile` (the one containing rules to build `matmul`) such that when `make team` is executed, all the team members' info is printed on the terminal (one member detail per line) in the format: <Name> <email ID> e.g. Raj Kumar 100200007@iitdh.ac.in **(0.5 points)**
2) Measure the total cycles consumed (using PAPI) for each configuration of the `matmul` run that you reported in PA1 report (Update your PA1 report with numbers from 18 distinct configurations/runs. (**4.5 points)**

OR

3) On `cares2`, list all the supported counters from which you can read the values using PAPI. **(5 points)**