End Term Examination
CS 2400

Total points – 70 (+ 5 bonus) points 29th November, 2017

Instructions

1. *Please READ ALL THE INSTRUCTIONS before proceeding to the questions.*
2. *This is an* **open notes** *examination. Photocopy of handwritten notes, electronic devices or other materials are not permitted.* **Any unethical behaviour in the examination may attract an 'FR' grade**.
3. *Answer questions in any order but all the subquestions has to be together. Start a* **new question** *on a* **new page**.
4. *Please be* very specific *and* very precise *in your answers.*
5. **(x page)** *next to questions indicate approximate length of an ideal answer with a normal font size and spacing. It is provided to you as a guideline to let you know that examiner* **doesn't expect longer answers**. *Don't try to forcefully squeeze your answer in this limit. There is no penalty if you use more space. But for the sanity of the examiner, please be brief and to the point.*
6. *Write legibly. Note that the burden is on you to make your answers easy to read as well as convincing and satisfactory to the examiner.*
7. *No clarifications will be provided.*
8. *Don't Panic.*

——— *Good Luck* ———

1. Consider Fig. 1a. Assume that all the elements of the array are unique. Provide the strongest possible loop variant for the code fragment. Assume that `N > 1 && N < INT_MAX` and size of the array `A` is `N`. Assume the type of other variables is `int`. (0.2 page) (10 points)

$$(\forall_{i1} : (0 \leq i1 < i) \rightarrow A[N-1-i1] > A[i1]) \bigwedge (\forall_{j1} : (N-1 \geq j1 > j) \rightarrow A[N-1-j1] < A[j1])$$

2. Look at the following function signature in C.

```
void substr(char* source,char* dest,int startpos,int length)
```

   (a) Provide precondition and postcondition for the function that should extract a substring from source from position `startpos` of `length` and copy the characters in the destination. (0.3 page) (20 points)
   Precondition :

$$\begin{aligned}&(length > 0 \wedge startpos \geq 0)\\ &\quad \wedge (\exists j \forall i : (startpos + length \leq j) \wedge (*(source + j) = NULL)\\ &\quad\quad \wedge ((0 \leq i < j) \rightarrow *(source + i) \neq NULL)\\ &\quad\quad\quad \wedge ((source + j < dest) \vee (dest + length < source)))\end{aligned}$$

```
for(i=0,j=N-1;i<j;i++,j--)
{ if(A[i]>A[j])
  {
    tmp=A[i];
    A[i]=A[j];
    A[j]=tmp;
  }
}
```

(a)

```
float sqrt (float num)
{
  float guess=1.0;
  while (guess*guess !=num )
   guess=(guess+num/guess)/2;
}
```

(b)

```
do{

 S1;
  if(e1) break;
 S2;
   if(e2) break;
 S3;
   if(e3) break;
 S4;
}
while(condition);
```

(c)

```
#include <iostream>
int dosomething(int a, int b=5)
{ return a+b; }
int dosomething(int a)
{ return a*5; }
int main()
{
std::cout << dosomething(5);
return 0;
}
```

(d)

```
int main() {
int x = 10, z=5;
int *p = &z;
int y;
x=45;
*p=15;
y = x/*p; /* what is the value of y? */;
return 0;
}
```

(e)

```
class A
{
  public:
    void method1(int param1) { body; };
    void method2(float param2) { body;};
    invariant(expr);
}
```

(f)

Figure 1: Program fragments

Postcondition:

$$
\begin{aligned}
&(\text{old\_source} = \text{source}) \\
&\qquad\wedge (\exists j \forall i \forall k : (\text{startpos} + \text{old\_length} \le j) \wedge (*(\text{old\_source} + j) = \text{NULL}) \\
&\qquad\qquad \wedge ((0 \le i \le j) \rightarrow *(\text{source} + i) = *(\text{old\_source} + i)) \\
&\wedge((0 \le k < \text{old\_length}) \rightarrow (*(\text{dest} + k) = *(\text{old\_source} + \text{startpos} + k)) \wedge (*(\text{dest} + k + 1) = \text{NULL})))
\end{aligned}
$$

(b) Change the signature to make it safer. (0.1 page) (5 points) `void substr(const char* source, char* dest, unsigned startpos, unsigned length)`

3. (a) Provide correct implementation to compute statistical mean with the function signature

`int mean(int a,int b)`

assuming that both a and b will be even or both will be odd. No typecast is allowed. (0.3 page) (5 points)

```
int mean(int a, int b)
 {
    if ((a%2==0) || (a<0&&b>0)|| (b<0&&a>0)) {return (a/2)+(b/2);}
    else if (a<0) {return (a/2)+(b/2)-1; }
```

```
            else {return (a/2)+(b/2)+1;}

      }
```

(b) Explain what would be the value of y before the function returns in Fig. 1e? (0.2 page) (5 points) Value of y would be 45 as the comment starts immediately after x.

(c) Consider Fig. 1b where sqrt is supposed to calculate square root. Assume num > 1.0. Explain the problem with the subroutine. Suggest a fix. (0.2 page) (5 points)

The subroutine uses equality operator with floating point numbers. The loop may not terminate at all. Check if the difference between guess*guess and num is less than some very small value, say 0.000001.

(d) Consider Fig. 1c. Rewrite an equivalent program fragment using a for loop, without introducing any extra variables. (0.3 page) (5 points)

```
 S1;
  if(!e1) {
 S2;
   if(!e2) {
 S3;
   if(!e3) {
 S4;

for(;condition;) {
 S1;
  if(e1) break;
 S2;
   if(e2) break;
 S3;
   if(e3) break;
 S4;
}
} } }
```

(e) Explain what would be the output in Fig. 1d. (0.3 page) (5 points) Compiler error due to ambiguity in deciding which function would be called when only one of the arguments is supplied.

4. What is a memory leak? How does a garbage collector prevent it? Explain with an example. (0.7 page) (10 points)

5. C++ doesn't have built-in support for object invariants. Rewrite class definition in Fig. 1f so that the invariant is enforced even without in-built support. (0.3 page) (5 bonus points) Put assert(expr); before and after the body in both the methods. All object invariants are precondition and postcondition to all member methods except for constructors and destructors. For constructor the invariant is just a postcondition. For destructor the invariant is just a precondition.

The End