CPSC 213

PRACTICE QUESTION SET 1

Question 1

2 local i and s and global n-> return a val

Here is a sm213 implementation of a procedure `foo` that takes no arguments and returns a value. Your goal is to write this procedure in C. The procedure uses the values of two local variables `i` and `s` which are kept in registers r1 and r2 and a global variable `n`. It may help if you first place some useful comments on the side of the assembly statements and then start the translation into C.

return a val

```
foo:    ld $n, r1              #
        ld 0x0(r1), r1         #
        ld $0x0, r2            #
loc1:   mov r1, r3             #
        not r3                 #
        inc r3                 #
        inc r3                 #
        bgt r3, loc2           #
        mov r1, r3             #
        shl $2, r3             #
        add r3, r2             #
        dec r1                 #
        br loc1                #
loc2:   mov r2, r0             #
        j 0x0(r6)              #
```

Question 2

Here is a sm213 implementation of a method `foo` of a Java class `Barfoo` that takes no arguments and returns a value. Your goal is to write this method in Java. The method uses the values of two local variables `i` and `s` which are kept in registers r1 and r2, and two instance variables `a` and `k` of the class `Barfoo`. You may assume that the `a` and `k` are the only instance variables of this class and that are allocated by the compiler in that order. First you should place some useful comments on the side of the assembly statements and then start the translation into Java.

```
foo:    deca r5              #

        st r4, 0x0(r5)       #

        ld 0x4(r7), r0       #

        ld $0x0, r1          #

        ld $0x0, r2          #

        ld 0x8(r7), r3       #

        not r3               #

        inc r3               #

        inc r3               #

loc1:   mov r1, r4           #

        add r3, r4           #

        bgt r4, loc2         #

        ld (r0,r1,4), r4     #

        shl $2, r4           #

        add r4, r2           #

        inc r1               #

        br loc1              #

loc2:   mov r2, r0           #

        ld 0x0(r5), r4       #

        inca r5              #

        j 0x0(r6)            #
```

Question 3

Consider the following Java code:

```
public class Foo {
     public static int i;
     private int a;
     private int j;
     private int k;

     public static int sbar() {
          . . .
     }

     public int bar() {
          . . .
     }

     public void foo () {
          i=0;
          j=0;
          k=0;
          i = sbar();
          a = bar();

     }
}
```

a. Give the sm213 Assembly code for thefirst three java instructions in `foo` which assign each of the variables i, j and k the value 0. Use the conventions we follow in this course about local variables, parameters and the use of registers r0, r5, r6 and r7. Clearly state any other assumptions you make.

b. Now give the sm213 Assembly code for the complete code of method foo.

Question 4

Consider the following Java code:

```
public class Foo {
     public static int i;
     private int a;
     private int j;
     private int k;

     public int bar1(int x) {
          . . .
     }

     public int bar2() {
          . . .
     }

     public void foo( int m, int n) {
          a = bar2();
          j = bar1(m);
          i=n;
          k=i+j;
     }
}
```

Give the sm213 Assembly code for the method foo. Assume r6 stores the return address, r7 stores the address of the current object and the parameters of foo are stored in registers r1 and r2. Don't worry about the old value of r6 when the function calls another function. Clearly state any other assumptions you make.

Question5

Memory Management, Pointers. Consider this C code.

```
#include <stdlib.h>
#include <stdio.h>

typedef struct {
     int n;
     int* darr;
} MyStruct;

/* Creates a new structure */
```

```
MyStruct* createMyStruct( int m ) {
     MyStruct* s = (MyStruct*) malloc( sizeof(MyStruct));
     s->n = m;
     s->darr = (int*) malloc(m+sizeof(int));
     return s;
}

/* sets the ith value in the structure s to the given value */
void setVal(MyStruct s, int i, int val){
     if (i<0 || i>=s.n)
          return;
     s.darr[i] = val;
}

/* Returns the address of the i-th value in structure s */
int* getVal(MyStruct s, int i){
     if (i<0 || i>=s.n)
          return NULL;
     return (s.darr + i);
}


main(){
     int i = 5;
     MyStruct* s = createMyStruct(i);
     int j;
     for (j = 0; j < i; j++) {
          setVal(*s, j, j+1000);
     }

     printf("Printing s: \n");
     for (j = 0; j < i; j++) {
          printf("Location %d has value %d \n", j, *getVal(*s, j));
     }

     MyStruct* t = createMyStruct(2*i);
     t = s;
     free(s);
     printf("Printing t: \n");
     for (j = 0; j < t->n; j++) {
          printf("Location %d has value %d \n", j, *getVal(*t, j));
     }

     free(t);

}
```

a. What does this code print after the call to main() ?

b. State whether this code has a dangling pointer bug, a memory leak bug, both bugs, or no memory allocation bugs. Concisely describe the exact problem

c. If you identified any bugs, state how to rewrite the code in order to fix them.

Question 6

Consider the following C code:

```
int d;

int proc1(int a) {
     int b = 10;
     d = a+b;
     return d;
}

void main() {
     proc1(5);
}
```

Provide the assembly for this code. Use the conventions we use in this course for the registers, procedure parameters and local variables .

Question 7

Consider the following Java code:

```
public class Bar {

     public static int i;

     private int j;
```

```
    static void bar(int m) {
        int k;
        i=0;
        m=0;
        j=0;
        k=0;
        return k;
    }
}
```

Give the sm213 Assembly code for the method bar of the class Bar. Assume r5 stores the stack pointer, r6 stores the return address and r7 stores the address of the current object. Clearly state any other assumptions you make.