CISS245: Advanced Programming
Test t01

Name: crsmith14@cougars.ccis.edu Score:

Open `main.tex` and enter answers (look for `answercode`, `answerbox`, `answerlong`). Turn the page for detailed instructions. To rebuild and view pdf, in bash shell execute `make`. To build a gzip-tar file, in bash shell execute `make s` and you'll get `submit.tar.gz`.

Instructions

- This is a closed-book, no-discussion, no-calculator, no-browsing-on-the-web no-compiler/no-MIPS-simulator test.
- Cheating is a serious academic offense. If caught you will receive an immediate score of -100%.
- If a question asks for a program output and the program or code fragment contains an error, write `ERROR` as output. When writing output, whitespace is significant.
- If a question asks for the computation of a value and the program or code fragment contains an error, write `ERROR` as value.
- When you're asked to write a C++ statement, don't forget that it must end with a semicolon.
- Bubblesort refers to the bubblesort algorithm in our notes where values are sorted in ascending order.

Honor Statement

I, Corban Rei Smith, attest to the fact that the submitted work is my own and is not the result of plagiarism. Furthermore, I have not aided another student in the act of plagiarism.

Question	Points
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Question	Points
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	

TOTAL	

Q1. Complete the following program. You can only add to the given code. You cannot delete or change what is given below.

Answer:

```
#include <iostream>

void swap(int * p, int * q)
{
  int t = *p;
  *p = *q;
  *q = t;
}


int main()
{
    int x = 0;
    int y = 1;

    swap(&x,&y);

    std::cout << x << ' ' << y << '\n'; // expected output: 1 0

    return 0;
}
```

Q2. Complete the following program. You can only add to the given code. You cannot delete or change what is given below.

The function `reverse(x, x_len)` reverses the values in the integer array `x` from index `0` up to index `x_len - 1`. For instance suppose the user enters `2 3 5 7 -9999`, then the values `2, 3, 5, 7` are placed in `x` and `x_len` is set to 4. After calling `reverse(x, x_len)`, the first four value in `x` becomes `7, 5, 3, 2`.

Answer:

```
#include <iostream>

void reverse(int x[], int x_len)
{

  for (int i = x[] - 2; i >= 0; --i)
  {
    for (int j = 0; j <= i; ++j)
    {
      if (x[j] > x[j + 1])
      {
        int t = x[j]; x[j] = x[j + 1]; x[j + 1] = t;
      }
    }

}


int main()
{
    int x[1024];
    int x_len;

    x_len = 0;
    for (int i = 0; i < 1024; ++i)
    {
        int t;
        std::cin >> t;
        if (t == -9999)
        {
            break;
        }
        x[x_len] = t;
        ++x_len;
    }

    reverse(x, x_len);
    // x from index 0 to x_len - 1 is now reverse.
```

```
    return 0;
}
```

Q3. What is the output of this program?

```
int x = 10;
int y = 20;
int * p = &x;
int * q = &y;
*p = *p * *q;
*q = 42;
std::cout << x << ' ' << y << '\n';
std::cout << *p << ' ' << *q << '\n';
```

Answer:

```
200 42
200 42
```

Q4. Rewrite the following code fragment by doing the following: Do a simple addition program using the following skeleton. You MUST follow the given instructions.

Answer:

```
int x = 0, y = 0;
int * p = &x;
int * q = &y;
// Do NOT declare any variables below.

// Statement to get an integer from user and put it into x.
// Do NOT use x in your statement.
std::cin >> *p;



// Statement to get an integer from user and put it into y.
// Do NOT use y in your statement.
std::cin >> *q;



std::cout << x + y << '\n';
```

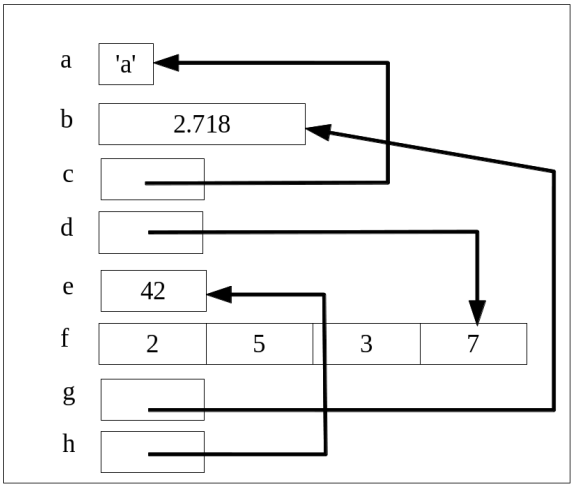
Q5. What is the output of this code fragment?

```
int i = 1;
int j = 2;
int k = 3;
int * p = &i;
int * q = &j;
int * r = &k;
p = q;
q = r;
std::cout << *p << ' ' << *q << ' ' << *r << '\n';
```

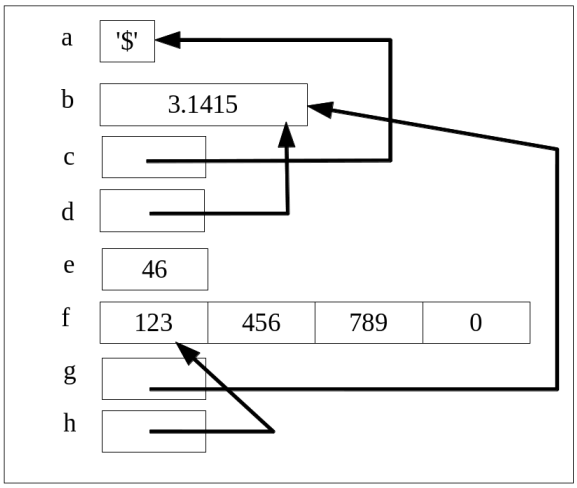
Answer:

```
2 3 3
```

Q6. You were brainstorming with your team in one of the company's meeting rooms. Your boss popped in to say hi on his way to get coffee and he noticed the following diagram on the whiteboard. Someone was tracing a piece of code on the whiteboard:



On his way back, your boss glanced at the whiteboard and saw this:



You noticed he was shaking his head as he walked away. Why?

Answer:

```
d went from int pointer to double pointer
d and g are both pointing to b
```

Q7. The following program does compile and does run. But it has a memory leak. Fix it so that there is no memory leak.

Answer:

```
#include <iostream>

int sum(int n)
{
    int s = 0;
    int * i = new int;
    for (*i = 0; *i <= n; ++(*i))
    {
      s += *i;
    }
    delete i;
    return s;
}

int main()
{
    std::cout << sum(10) << '\n';
    return 0;
}
```

Q8. The following program get two integer values from the user and then prints the sum. Do NOT use integer or double variables – you can only use pointers. In fact I have already declared all the variables you need, i.e., two pointer variables. You must allocate and deallocate memory correctly.

Answer:

```
#include <iostream>

int main()
{
    int * p;
    int * q;

    // allocate memory for p
    int * p = new int;



    // allocate memory for q
    int *q = new int;

    // get integer value from user and store at integer that p points to
    std::cin >> *p;

    // get integer value from user and store at integer that q points to
    std::cin >> *q;

    // print the sum of integers that p and q point to
    std::cout << *p + *q << '\n';

    // deallocate memory used by q
    delete q;

    // deallcate memory used by p
    delete p;

    return 0;
}
```

Q9. Complete this code segment.

Answer:

```
int x[] = {1, 5, 3, 7, 9, 4, 2, 6, 8, 0};
int max;

int * start = &x[0];
int * end = &x[10];
int * p;
int * pmax = &max;

// Complete the following to compute the maximum value in the array x
// from *start to *(end - 1) and store it in variable max.
// Your code must work for different array values in x.
// You also cannot use the name x or max.
// You must use a loop (of course).
// You can only use integer pointer variables start, end, p, pmax
void bubblesort(int * start,int  * end, int * p, int * pmax)
{
  *p = &x[0];
    for (int * p = end - 1; p >= start; --p)
    {
        for (int * q = start; q <= p; ++q)
        {
            if (*q > *(q + 1))
            {
                swap(*q, *(q + 1));
            }
        }
        if (*q > *p)
        {
          *pmax = *q;
        }
    }
}




// At this point the maximum value of *start,...,*(end - 1) is stored in
// variable max.
std::cout << max << '\n';
```

Q10. The following have a function that attempts to perform memory allocation and memory deallocation, but they are done incorrectly:

```
void mynew(int * p)
{
    p = new int;
}

void mydelete(int * p)
{
    delete p;
    p = NULL;
}

int main()
{
    int * p;
    mynew(p);
    *p = 42;
    mydelete(p);
    return 0;
}
```

Fix the above problem below.

Answer:

```
void mynew(int *p)
{
  p = new int;
}

void mydelete(int * p)
{
  delete p;
}

int main()
{
    int * p;
    mynew(p);
    *p = 42;
    mydelete(p);
    return 0;
}
```

Q11. Complete the following by writing a struct and making any corrections.
Answer:

```
#include <iostream>

// define the struct here

struct student
{
  id = x.student_id;
  year = x.dob_year;
  month = x.dob_month;
  day = x.dob_day;
  height = x.height;
  weight = x.weight;
}



void input(Student * x)
{
    std::cin >> x.student_id; // get an integer value from user for x's
                              // student id
    std::cin >> x.dob_year;   // get an integer value from user for x's year
                              // of date of birth
    std::cin >> x.dob_month;  // get an integer value from user for x's month
                              // of date of birth
    std::cin >> x.dob_day;    // get an integer value from user for x's day
                              // of date of birth
    std::cin >> x.height;     // get a double value from user for x's height
    std::cin >> x.weight;     // get a double value frin user for x's weight
}

void print(Student & x)
{

  std::cout << student_id << ',' << student_year << ',' << student_month << ','
  << student_day << ',' << student_height << ',' << student_weight << '\n';




}

int main()
{
    Student john;
    input(john);
    println(john); // print all values of john separated by ',' and print '\n'
    return 0;
```

```
}
```

Q12. You are writing a tic-tac-toe game. The following code is in your `main()`:

```
#include <iostream>
#include "TTT.h"

int main()
{
    TTT board;

    while (1)
    {
        print(board);
        int row, col;
        get_input(board, row, col);
        make_move(board, row, col);
        if (game_ended(board))
        {
            break;
        }
    }
    print_result(board);

    return 0;
}
```

Complete the header file (with the struct definition and the function prototypes – no function body definitions). The struct and function prototypes must be minimal (i.e., no useless member variables, no unnecessary parameters, reference parameters must be constant whenever possible).

Answer:

```
#ifndef TTT_H
#define TTT_H


  struct board
  {



  }
```

```
#endif
```

Q13. What is the output? Or is there an error?

```
#include <iostream>

int h(int * p)
{
    return *p;
}

int * g(int * p)
{
    return p;
}

int * f(int * p)
{
    return (p != NULL ? g(p) : NULL);
}

int main()
{
    int i = 5;
    std::cout << *f(&i) + h(&i) << std::endl;
    return 0;
}
```

Answer:

```
error because h is not a pointer
```

Q14. Complete the following program. Make sure there is no memory leak.

Answer:

```
#include <iostream>

int f(int n)
{
    int * p;

    // Allocate an integer array of size n to p. (Of course the array
    // is in the heap.)
    int n [] = *p;




    // Fill the array that p points to with values 1, 2, 3, ..., n.
    for (int i = 0; i < 3; ++i)
    {
      n[] = i;
    }

    // Go over the values in the array that p points to and
    // (1) if a value is odd, replace that value by the square root of the
    //     value, or
    if (n % 3 || n == 1)
    {
      n *= n;
    }

    // (2) if a value x is even, replace that value x by x + 1.
    else
    {
      n += 1;
    }
    // This is one pass.
    // Repeat this until every value in the array is <= 42.
    // Return the number of passes you have to run over the array

    int ret; // number of passes
    n = *p;
    int x = [n];
    for (int i = 0 ; i <=42; ++ i)
    {
      for (int i = 0; i < 3; ++i)
      {
        n[] = i;
      }
      if (n % 3 || n == 1)
```

```
      {
        n *= n;
      }
      else
      {
        n += 1;
      }
      ++ ret;

    }
    std::cout << ret << '\n';
```

```
    return ret;
}

int main()
{
    int n;
```

```
    std::cin >> n;
    std::cout << f(n) << '\n';
    return 0;
}
```

Q15. Complete the following function that performs the binary search. You need NOT use recursion.

Answer:

```
// Performs binary search on *start, *(start+1), ..., *(end - 1) for the
// value of target and return the pointer where target is found.
// If target is not found, NULL is returned.
int * binarysearch(int * start, int * end, int target)
{
  int * lower = start;
  int * upper = end - 1;
  while (lower <= upper)
  {
    int * mid = lower + (upper - lower) / 2;
        if (*mid == target)
        {
          return mid;
        }
        else if (target < *mid)
        {
          upper = mid - 1;
        }
        else
        {
          lower = mid + 1;
        }
  }
  return NULL;
}
```

Q16. Complete the program below. Here are two test cases.

Test 1

```
1 2 3
4 5 6
4 5 6
1 2 3
```

Test 2

```
10 9 8
5 6 7
5 6 7
10 9 8
```

Answer:

```
#include <iostream>

void swap(int ** p, int ** q))
{
    // You can only write at most 3 statements
    int t = &p;
    &p = &q ;
    &q = t

    return;
}

int main()
{
    int * p = new int[3];
    int * q = new int[3];
    for (int i = 0; i < 3; ++i)
    {
        std::cin >> p[i];
    }
    for (int i = 0; i < 3; ++i)
    {
        std::cin >> q[i];
    }

    swap(&p, &q);
    for (int i = 0; i < 3; ++i)
    {
        std::cout << p[i] << ' ';
    }
    std::cout << '\n';
    for (int i = 0; i < 3; ++i)
    {
```

```
        std::cin << q[i] << ' ';
    }
    std::cout << '\n';
    delete [] p;
    delete [] q;
    return 0;
}
```

Instructions

In `main.tex` change the email address in

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

yours. In the bash shell, execute "`make`" to recompile `main.pdf`. Execute "`make v`" to view `main.pdf`. Execute "`make s`" to create `submit.tar.gz` for submission.

For each question, you'll see boxes for you to fill. You write your answers in `main.tex` file. For small boxes, if you see

```
1 + 1 = \answerbox{}.
```

you do this:

```
1 + 1 = \answerbox{2}.
```

`answerbox` will also appear in "true/false" and "multiple-choice" questions.

For longer answers that needs typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
int x;
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?
\begin{answerlong}
\end{answerlong}
```

you can write

```
What is the color of the sky?
\begin{answerlong}
The color of the sky is blue.
\end{answerlong}
```

For students beyond 245: You can put LaTeX commands in `answerlong`.

A question that begins with "T or F or M" requires you to identify whether it is true or false, or meaningless. "Meaningless" means something's wrong with the statement and it is not well-defined. Something like "$1+_2$" or "$\{2\}^{\{3\}}$" is not well-defined. Therefore a question such as "Is $42 = 1+_2$ true or false?" or "Is $42 = \{2\}^{\{3\}}$ true or false?" does not make sense. "Is $P(42) = \{42\}$ true or false?" is meaningless because $P(X)$ is only defined if X is a set. For "Is $1 + 2 + 3$ true or false?", "$1 + 2 + 3$" is well–defined but as a "numerical expression", not as a "proposition", i.e., it cannot be true or false. Therefore "Is $1 + 2 + 3$ true or false?" is also not a well-defined question.

When writing results of computations, make sure it's simplified. For instance write 2 instead of $1 + 1$. When you write down sets, if the answer is $\{1\}$, I do not want to see $\{1, 1\}$.

When writing a counterexample, always write the simplest.

Here are some examples (see `instructions.tex` for details):

1. T or F or M: $1 + 1 = 2$ ..... T
2. T or F or M: $1 + 1 = 3$ ..... F
3. T or F or M: $1+^2 =$ ..... M
4. $1 + 2 =$ 3
5. Write a C++ statement to declare an integer variable named `x`.

   ```
   int x;
   ```

6. Solve $x^2 - 1 = 0$.

   Since $x^2 - 1 = (x - 1)(x + 1)$, $x^2 - 1 = 0$ implies $(x - 1)(x + 1) = 0$. Therefore $x - 1 = 0$ or $x = -1$. Hence $x = 1$ or $x = -1$.

7. Which is true? C
   (A) $1 + 1 = 0$
   (B) $1 + 1 = 1$
   (C) $1 + 1 = 2$
   (D) $1 + 1 = 3$
   (E) $1 + 1 = 4$