COMP208 FINAL REVIEW

EPTS COMP Tutor Shabbir Hussain

Useful Links

Course Summary:

- http://s3.amazonaws.com/docuum/attachments/2086/comp%20208%20info.pdf?1240285685

Run C code online:

- http://codepad.org/
- http://www.compileonline.com/compile_c_online.php

Run Fortran code online:

- http://www.compileonline.com/compile_fortran_online.php

Study Strategies

1. Do past finals
    a. Check the course website
    b. Start by doing the hardest questions first
    c. Test your code on the computer
2. Make a cheat sheet (as an exercise)

Contents

```c
#include <stdio.h>
#include <string.h>

main()
{

    int i=10;
    int j=i;
    int* p = &i;
    int* q = &p;

    *p += 10;
    *q = &j;
    *p += 10 + *p;

    printf("%d %d\n", i,j);

    return 0;

```

What will be printed if we run this program?

Solve this problem as if you’re executing the code like a computer

Disclaimer: the techniques shown are not necessarily how the computer does it, but how you should do it on an exam

Pointers Example

```
1   #include <stdio.h>
2   #include <string.h>
3
4   main()
5   {
6
7       int i=10;
8       int j=i;
9       int* p = &i;
10      int* q = &p;
11
12      *p += 10;
13      *q = &j;
14      *p += 10 + *p;
15
16      printf("%d %d\n", i,j);
17
18      return 0;
19
20  }
```

line 1 to 10 is just declarations. Run till line 10 and build a table of variables:

Name	Value	Addr
i	10	&i
j	10	&j
p	&i	&p
q	&p	&q

Pointers Example

```
1  #include <stdio.h>
2  #include <string.h>
3
4  main()
5  {
6
7      int i=10;
8      int j=i;
9      int* p = &i;
10     int* q = &p;
11
12     *p += 10;
13     *q = &j;
14     *p += 10 + *p;
15
16     printf("%d %d\n", i,j);
17
18     return 0;
19
20 }
```

```
*p += 10;
```

```
i += 10;
```

Line 12 is the first line that edits a value

Anytime you see a dereference. Check the table and substitute the name of the correct variable.

Name	Value	Addr
i	10	&i
j	10	&j
p	&i	&p
q	&p	&q

Pointers Example

```c
#include <stdio.h>
#include <string.h>

main()
{

    int i=10;
    int j=i;
    int* p = &i;
    int* q = &p;

    *p += 10;
    *q = &j;
    *p += 10 + *p;

    printf("%d %d\n", i,j);

    return 0;

}
```

```c
*p += 10;
```

```c
i += 10;
```

Now update the value in the table

Name	Value	Addr
i	20	&i
j	10	&j
p	&i	&p
q	&p	&q

Pointers Example

```c
#include <stdio.h>
#include <string.h>

main()
{

    int i=10;
    int j=i;
    int* p = &i;
    int* q = &p;

    *p += 10;
    *q = &j;
    *p += 10 + *p;

    printf("%d %d\n", i,j);

    return 0;

}
```

```c
*q = &j;
```

```c
p = &j;
```

Move to the next Line and do the same procedure:

1. Substitute with correct name
2. update value

Name	Value	Addr
i	20	&i
j	10	&j
p	&j	&p
q	&p	&q

Pointers Example

```c
#include <stdio.h>
#include <string.h>

main()
{

    int i=10;
    int j=i;
    int* p = &i;
    int* q = &p;

    *p += 10;
    *q = &j;
    *p += 10 + *p;

    printf("%d %d\n", i,j);

    return 0;

}
```

```c
*p += 10 + *p;
```

```c
j += 10 + j;
```

Move to the next Line and do the same procedure:

1. Substitute with correct name
2. update value

Name	Value	Addr
i	20	&i
j	30	&j
p	&j	&p
q	&p	&q

Pointers Example

```c
#include <stdio.h>
#include <string.h>

main()
{

    int i=10;
    int j=i;
    int* p = &i;
    int* q = &p;

    *p += 10;
    *q = &j;
    *p += 10 + *p;

    printf("%d %d\n", i,j);

    return 0;

}
```

Executing the program....

$demo

```
20 30
```

The program will now print the values of i and j

Name	Value	Addr
i	20	&i
j	30	&j
p	&j	&p
q	&p	&q

Pointers more Examples

```
#include <stdio.h>

main()
{
    int a[10];
    int *p =a;
    int i;
    for(i=0; i<10; i++){
        a[i]=10-i;
    }

    printf("%d %d %d %d\n",a[3],*p,p[4],*a+5);

   return 0;
}
```

What is the output of this program?

Build a table after all values are initialized

Pointers more Examples

```
#include <stdio.h>

main()
{
    int a[10];
    int *p =a;
    int i;
    for(i=0; i<10; i++){
        a[i]=10-i;
    }

    printf("%d %d %d %d\n",a[3],*p,p[4],*a+5);

   return 0;
}
```

Name	Value	address
a[0]	10	a
a[1]	9	a+1
a[2]	8	a+2
a[3]	7	a+3
a[4]	6	a+4
a[5]	5	a+5
a[6]	4	a+6
a[7]	3	a+7
a[8]	2	a+8
a[9]	1	a+9
p	a	&p

Pointers more Examples

```
//start with the original equation
printf("%d %d %d %d\n",a[3],*p,p[4],*a+5);

//use table to substitute correct variable name:
printf("%d %d %d %d\n",a[3],a[0],a[4],a[0]+5);

//use table to substitute correct values
printf("%d %d %d %d\n",7,10,6,15)
```

Name	Value	address
a[0]	10	a
a[1]	9	a+1
a[2]	8	a+2
a[3]	7	a+3
a[4]	6	a+4
a[5]	5	a+5
a[6]	4	a+6
a[7]	3	a+7
a[8]	2	a+8
a[9]	1	a+9
p	a	&p

Trick Questions (Don’t be fooled)

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

void abc(float, float, float);
main()
{

    float y =2.5;
    abc(6.5, y, y);

    printf("%f\n", y);

    return 0;

}

void abc(float x, float y, float z){

    y = y-1;
    z = z+x;
}
```

What is the Output of this program?

Hint: answer is on the next slide. Don’t look until you try it!

Taken from Fall 2007 Final

Trick Questions (Don’t be fooled)

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

void abc(float, float, float);
main()
{

    float y =2.5;
    abc(6.5, y, y);

    printf("%f\n", y);

    return 0;

}

void abc(float x, float y, float z){

    y = y-1;
    z = z+x;
}
```

Executing the program....
$demo

2.500000

Whenever there is a function check the types for the inputs and the outputs to determine if values are modified.

Here inputs are passed by value.

Trick: the value of y does not change

Trick Questions (Don’t be fooled)

```
program exam
IMPLICIT NONE
INTEGER :: array(5), i, k

Do i=1,5
    array(i) = i
END DO
DO k=5,1,-1
    array(k) = mod(array(i-1), array(k))
END DO

   WRITE (*,*) (array(i), i=1,5)

end program exam
```

What is the Output of this Program?

Taken from Fall 2006 Final

Trick Questions (Don’t be fooled)

```
program exam
IMPLICIT NONE
INTEGER :: array(5), i, k

Do i=1,5
    array(i) = i
    WRITE (*,*) (i)
END DO
    WRITE (*,*) (i)
DO k=5,1,-1
    array(k) = mod(array(i-1), array(k))
END DO

   WRITE (*,*) (array(i), i=1,5)

end program exam
```

Lets add a few more write statements so that we can understand what’s going on...

Executing the program....

```
$demo

1
2
3
4
5
6
0          0          0          0          0
```

Trick: variable i gets incremented to 6!

Trick Questions (Don’t be fooled)

General Solution to these types of problems:

1. Know your stuff
2. Relax, its not worth that much anyway

Big O

How does the Upper bound complexity grow?

Its a measure of either run-time or resources (memory slots, etc.)

Big O

Question 12

What is the complexity (big-Oh) of the following program segment?

```
for(i=0;i<n;++i)
   for(j=0;j<2*n;++j)
      for(k=1;k<3*n;++k)  ;
```

a) $O(n+2n+3n)$
b) $O(3n)$
c) $O(n^2)$
d) $O(n^3)$
e) $O(2^n)$

Taken from fall 2006 final

Big O

Question 12

What is the complexity (big-Oh) of the following program segment?

```
for(i=0;i<n;++i)            n
  for(j=0;j<2*n;++j)        2n
    for(k=1;k<3*n;++k) ;    3n-1
```

a) O(n+2n+3n)

b) O(3n)

c) $O(n^2)$

d) $O(n^3)$

e) $O(2^n)$

n*(2n)*(3n-1)
= 2n^2*(3n-1)
= 6n^3 - 2n^2

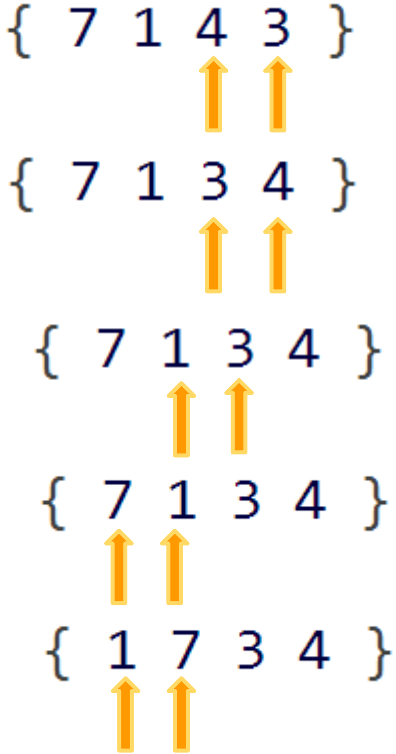
Take the highest order

Taken from fall 2006 final

Sorting Algorithms in Plain English

One pass of Bubble sort:

1. Start at the back
2. Swap if that element is less than the preceding
3. Move to the next element (continue till first element)



Sorting Algorithms in Plain English

One pass of Selection sort:

1. Find the smallest element
2. Swap it with the front

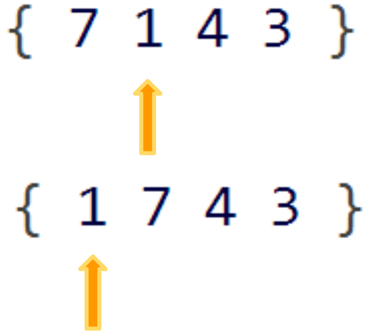
{ 7 1 4 3 }

{ 1 7 4 3 }

Sorting Algorithms in Plain English

One pass of Selection sort:

1. Find the smallest element
2. Swap it with the front



What would the array look like after a second pass?

Sorting Algorithms in Plain English

One pass of Selection sort:

1. Find the smallest element
2. Swap it with the front

{ 7 1 4 3 }

{ 1 7 4 3 }

What would the array look like after a second pass?

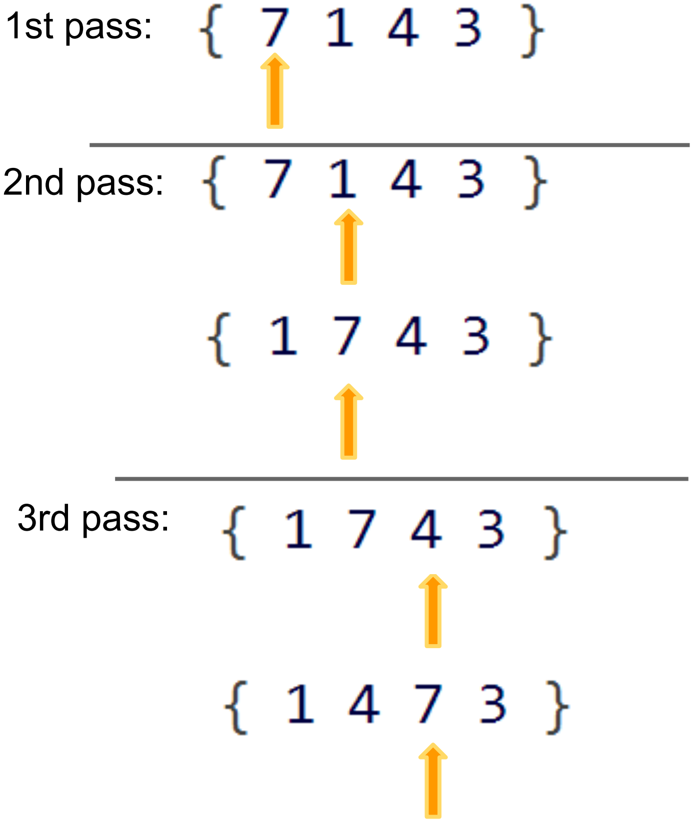
{ 1 7 4 3 }

{ 1 3 4 7 }

Sorting Algorithms in Plain English

One pass of Insertion sort:

1. Take an element from the rest of the list
2. Insert it to the sorted list



Sorting Pop Quiz

Question 11

If the bubble sort algorithm is applied to the array: {5,3,2,4,1}, what will be the arrangement of the elements after two passes?

a) {1,2,3,4,5}
b) {1,2,5,3,4}
c) {3,5,2,4,1}
d) {3,5,1,2,4}
e) None of the above

Taken from 2006 Fall Final

Sorting Pop Quiz

Question 11
If the bubble sort algorithm is applied to the array: {5,3,2,4,1}, what will be the arrangement of the elements after two passes?

a) {1,2,3,4,5}
b) {1,2,5,3,4}
c) {3,5,2,4,1}
d) {3,5,1,2,4}
e) None of the above

Taken from 2006 Fall Final

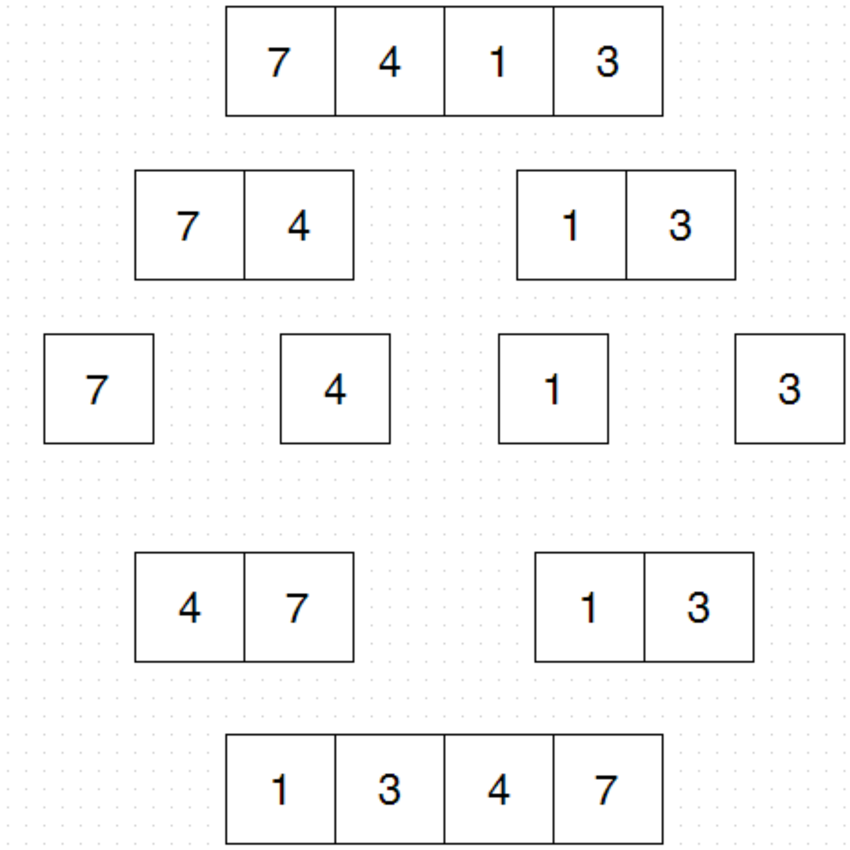
1st Pass: { 5 3 2 4 1 }
{ 5 3 2 1 4 }
{ 5 3 1 2 4 }
{ 5 1 3 2 4 }
{ 1 5 3 2 4 }

2nd Pass: { 1 5 3 2 4 }
{ 1 5 2 3 4 }
{ 1 2 5 3 4 }

Sorting Algorithms in Plain English

Merge sort:

1. If list has only one element you are done
2. otherwise separate it into two lists
3. merge sort on first half
4. merge sort on second half
5. recombine



Recursion

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

int f(int x, int n){
    if(n<4)
        return f(x+1,n+1) + f(x+2,n+1);
    else
        return x;
}

main()
{
   printf("%i\n", f(0,0));
   return 0;
}
```

What is the output of the following Program?

Hint: expand the function

Recursion

Expand the recurrence relation:

$f(x,4) = x$
$f(x,n)= f(x+1,n+1)+f(x+2,n+1)$

f(0,0)

= f(1,1) +f(2,1)

= f(2,2) +f(3,2) + f(3,2) +f(4,2)

= f(3,3) +f(4,3) +f(4,3) +f(5,3) +f(4,3) +f(5,3) +f(5,3) +f(6,3)

=f(4,4)+f(5,4)+f(5,4)+f(6,4)+f(5,4)+f(6,4)+f(6,4)+f(7,4)+f(5,4)+f(6,4)+f(6,4)+f(7,4)+f(6,4)+f(7,4)+f(7,4)+f(8,4)

= 4 +5 +5 + 6 + 5 + 6 + 6 +7 + 5 +6 +6 +7 +6 +7 +7 +8

=96

Numerical Methods

```
double findroot(DfD f, double x1, double x2,
                       double tol, int count)
{
      double f1 = f(x1), f2 = f(x2),
             slope = (f2 - f1) / (x2 - x1),
             distance = -f2 / slope,
             point = x2 + distance;
      if(!count)
           return point;
      if(fabs(f(point)) < tol)
           return point;
      return findroot(f, x2, point, tol, count - 1);
}
```

This is an implementation of the:

a) Bisection method
b) Secant method
c) False-position method
d) Newton-Raphson method
e) None of the above

Taken from Fall 2005 final

Numerical Methods (root finding)

Passing functions as Arguments

```
typedef double (*DfD) (double);
typedef double (*DfDD) (double, double);
typedef double (*DfDDD) (double, double, double);

double bisection_rf(DfD f, double x0, double x1, double tol);
```

f is of type DFD which is a function that takes a double as input and returns a double

Numerical Methods

```
double findroot(DfD f, double x1, double x2,
                       double tol, int count)
{
     double f1 = f(x1), f2 = f(x2),
            slope = (f2 - f1) / (x2 - x1),
            distance = -f2 / slope,
            point = x2 + distance;
     if(!count)
          return point;
     if(fabs(f(point)) < tol)
          return point;
     return findroot(f, x2, point, tol, count - 1);
}
```

This is an implementation of the:

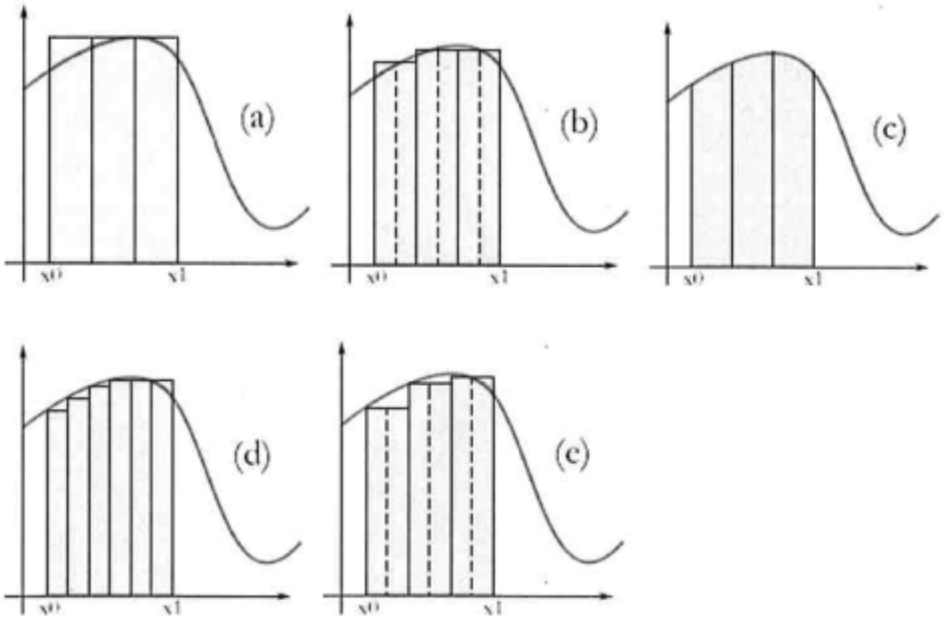
a) Bisection method
b) Secant method
c) False-position method
d) Newton-Raphson method
e) None of the above

How did I know? Because its the code copied from notes. Read your class notes!

Numerical Methods (Integration)

Question 7

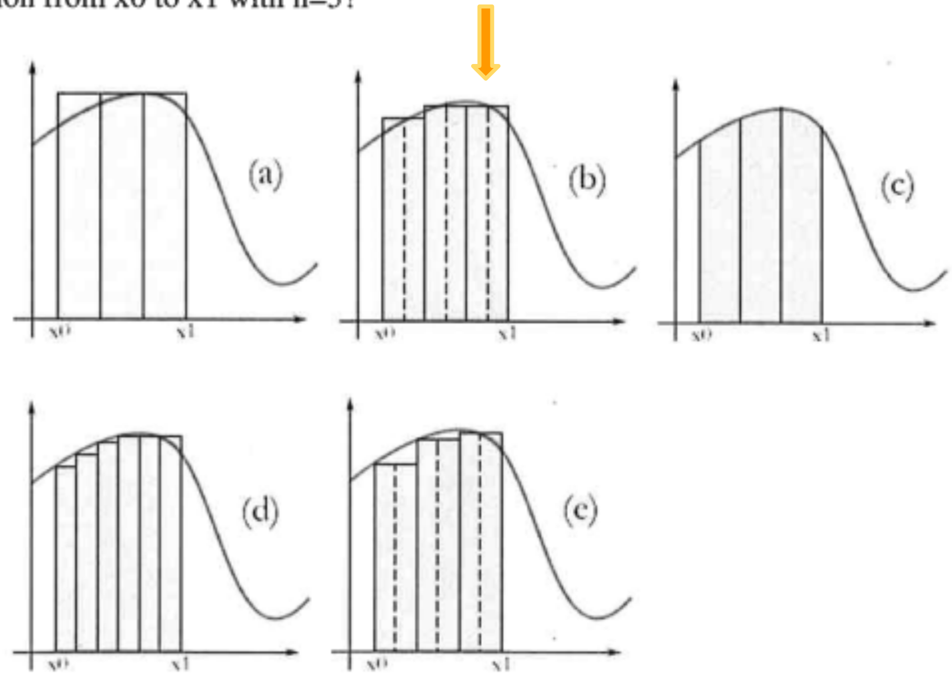
Which of these pictures best represents the mid-point integration algorithm used to integrate a function from x0 to x1 with n=3?



Taken from winter 2007 final

Numerical Methods (Integration)

Question 7
Which of these pictures best represents the mid-point integration algorithm used to integrate a function from x0 to x1 with n=3?



Mid point method evaluates the function at the middle of the interval

Taken from winter 2007 final

Numerical Methods (IVP)

Question 15

Which of the following statements about solving Initial Value Problems are true?

1. The Euler method can not be used to solve the equation: $dy/dx = x + y^3 + \log x$
2. The Runge-Kutta method is generally more accurate than the Euler method.
3. The Euler method needs an initial value to solve an ODE, but Runge-Kutta does not.
4. The Euler method does not give as accurate a result as the analytical solution, but the Runge-Kutta method does.

a) 1, 2
b) 2, 3
c) 3, 4
d) 2
e) None of the above.

Taken from Fall 2005 final

Numerical Methods (IVP)

Question 15
Which of the following statements about solving Initial Value Problems are true?

1. The Euler method can not be used to solve the equation: $dy/dx = x + y^3 + \log x$
2. The Runge-Kutta method is generally more accurate than the Euler method.
3. The Euler method needs an initial value to solve an ODE, but Runge-Kutta does not.
4. The Euler method does not give as accurate a result as the analytical solution, but the Runge-Kutta method does.

a) 1, 2
b) 2, 3
c) 3, 4
d) 2
e) None of the above.

1. false, Euler method is used for any first order ODE
2. true, euler method needs a smaller step size to achieve the same accuracy
3. false, these are all methods for solving initial value problems
4. false, All numerical methods have some error

Numerical Methods

Is there a trick to do these types of problems?

Yes. Memorize all the algorithms

Good Luck!