Name:_____ NetID:_____

Answer all questions in the space provided. Write clearly and legibly, you will not get credit for illegible or incomprehensible answers. This is a closed book exam. However, each student is allowed to bring one page of notes to the exam. Print your name at the top of every page.

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	9	18	9	12	12	12	10	10	8	100
Score:										

1. For the following questions, select the single best answer.

   (a) In the C Programming Language, *call by value* means: (3)

   A. When two functions have the same name, the compiler determines which to call by the value of the arguments.

   B. The called function is given the values of its arguments which are copied into temporary variables.

   C. The called function is given the address of its arguments so that the function can both read and set the argument's values.

   D. Each called function is assigned a value that is used by the operating system to determine the function's priority. This is most useful on multi-core systems.

   (b) In the C Programming Language, an *automatic variable* is: (3)

   A. A global variable that is automatically available to all functions within the source file.

   B. A global variable that is available to all functions within any source file that declare the variable as extern.

   C. A variable that is automatically defined by the compiler such as PI, E, and HBAR.

   D. A local variable in a function which comes into existence at the time the function is called, and disappears when the function is exited.

   E. A variable that is automatically initialized.

   (c) What is the final value of i when the following code is run? (3)

```
int i;
for(i=0; i<3; i++) { }
```

   A. 0

   B. 1

   C. 2

   D. 3

   E. 4

   F. undefined

2. For the following questions, select the single best answer.

(a) Which of the following gives the memory address of integer variable `a`? (3)

A. `addressof(a)`
B. `a`
C. `a[0]`
D. `&a`
E. `*a`

(b) Which of the following accesses a field in structure `b`? (3)

A. `b->var`
B. `b.var`
C. `b-var`
D. `b>var`

(c) Which of the following accesses a field in a structure pointed to by pointer `b`? (3)

A. `b->var`
B. `b.var`
C. `b-var`
D. `b>var`

(d) Which of the following is a properly defined struct? (3)

A.
```
struct {int a;}
```

B.
```
struct Foo {int a;}
```

C.
```
struct Foo int a;
```

D.
```
struct Foo {int a;};
```

E.
```
struct {Foo a;};
```

(e) Which of the following is the proper keyword or function to allocate memory in C? (3)

A. new
B. malloc
C. create
D. value

(f) Which of the following is the proper keyword or function to deallocate memory? (3)

A. free
B. delete
C. clear
D. remove

3. For the following questions, select the single best answer.

(a) Consider the following variable declarations. (3)

```
char c = 'm';
char* p = &c;
char** q = &p;
```

Which of the following lines of code will *not* result in `c` having a value of `'n'`?

A. `c = 'n';`
B. `c++;`
C. `*p += 1;`
D. `*p++;`
E. `++p[0];`
F. `++(**q);`
G. `*q[0] += 1;`
H. `q[0][0]++;`

(b) Consider the following code. (3)

```
1 void main(void)
2 {
3   long a[] = {7, 13, 17};
4   long *x = a;
5   printf("sizeof(long)=%lu ", sizeof(long));
6   printf("x=%p, x[0]=%ld\n", x, x[0]);
7   x = x + 2;
8   printf("x=%p, x[0]=%ld\n", x, x[0]);
9 }
```

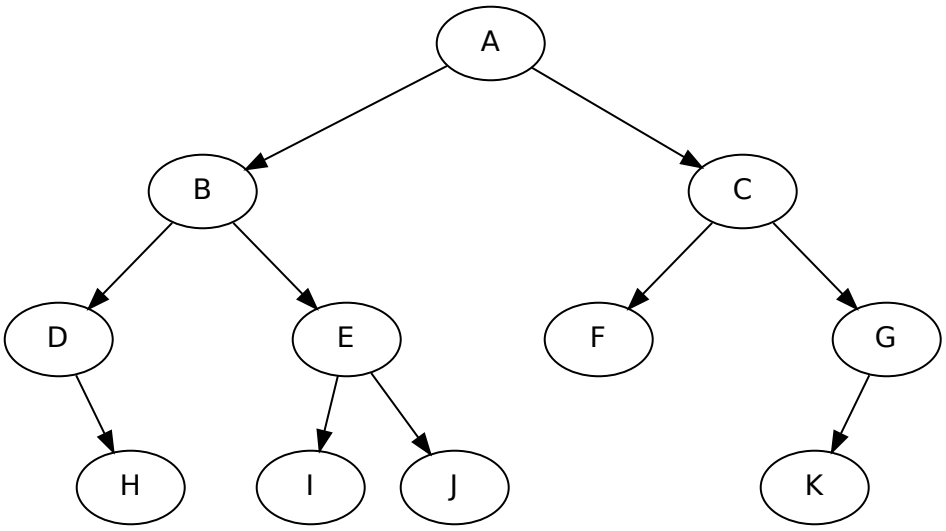
If the output from lines 5 and 6 is:
`sizeof(long)=8 x=0x7fff04794670, x[0]=7`
Then the output from line 8 will be:

A. `x=0x7fff04794672, x[0]=7`
B. `x=0x7fff04794672, x[0]=13`
C. `x=0x7fff04794672, x[0]=17`
D. `x=0x7fff04794678, x[0]=7`
E. `x=0x7fff04794678, x[0]=17`
F. `x=0x7fff04794680, x[0]=7`
G. `x=0x7fff04794680, x[0]=17`
H. `x=0x7fff04794686, x[0]=7`
I. `x=0x7fff04794686, x[0]=17`

(c) In what order do the two command line variables appear in the definition of `main`? (3)

A. Count then argument array
B. Argument array then count
C. There aren't any arguments in the definition of main.
D. There is only one argument, an array of strings.

4. Consider the following tree.



Write the number of the appropriate sequence for each traversal.

1. A, B, C, D, E, F, G, H, I, J, K
2. A, B, D, H, E, I, J, C, F, G, K
3. A, B, D, H, I, J, E, C, G, K, F
4. A, C, G, K, F, B, E, J, I, D, H
5. D, H, B, I, E, J, A, F, C, K, G
6. H, D, I, J, E, B, F, K, G, C, A
7. H, I, J, K, D, E, F, F, B, C, A
8. K, G, F, C, J, I, E, H, D, B, A

(a) Breadth First (also known as level-order): (3)

(a) _____

(b) Depth First, in-order: (3)

(b) _____

(c) Depth First, pre-order: (3)

(c) _____

(d) Depth First, post-order: (3)

(d) _____

5. The following program compiles and runs. What is its output? (12)

```
#include <stdio.h>

void main(void)
{
  char s[] = "QfiQQnQalQ";
  char del = 'Q';

  int sourceIndex = 0;
  int sinkIndex = 0;
  while (s[sourceIndex])
  {
    if (s[sourceIndex] != del)
    {
      s[sinkIndex] = s[sourceIndex];
      sinkIndex++;
    }
    else
    {
      printf("[%d,%d] %s\n", sourceIndex, sinkIndex, s);
    }
    sourceIndex++;
  }
  s[sinkIndex]='\0';
  printf("result: %s\n",s);
}
```

6. The following program compiles and runs. What is its output? (12)

   Hint: the `%x` format specifier prints an integer in hexadecimal format (with lowercase letters), the `#` says to prefix the result with `0x`, and the `04` says to make the output at least 4 characters wide, padding with leading zeros if needed. So, for example, the statement `printf("%#04x", 13);` would result in `0x0d` being printed to standard out.

```
#include <stdio.h>

void main(void)
{
  unsigned char x = 72;
  unsigned char y = 31;

  unsigned char a = x << 3;
  unsigned char b = x >> 3;
  unsigned char c = x & y;
  unsigned char d = x | y;
  unsigned char e = x ^ y;

  printf("x %#04x, %d\n", x, x);
  printf("y %#04x, %d\n", y, y);

  printf("a %#04x, %d\n", a, a);
  printf("b %#04x, %d\n", b, b);
  printf("c %#04x, %d\n", c, c);
  printf("d %#04x, %d\n", d, d);
  printf("e %#04x, %d\n", e, e);
}
```

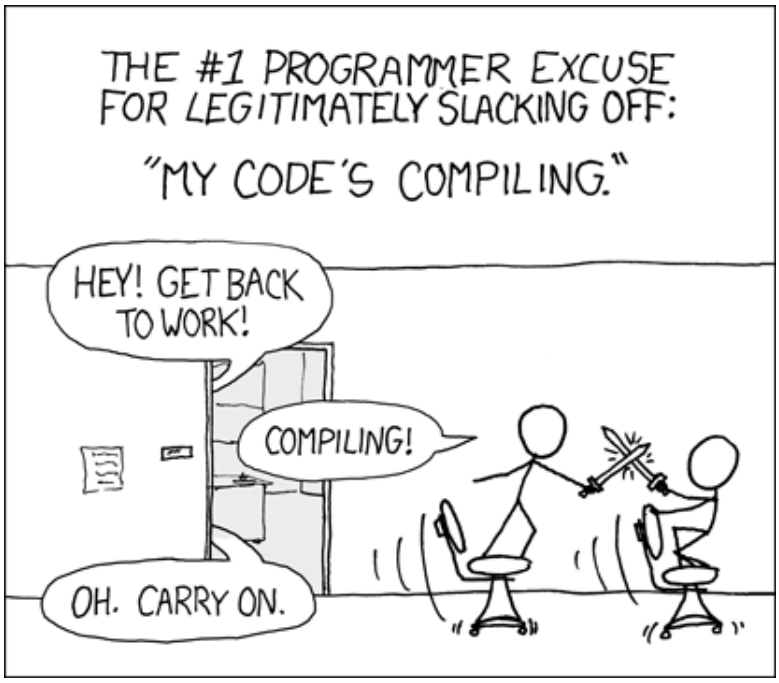
7. The following program compiles and runs. What is its output? (10)

```
#include <stdio.h>

int foo(int f[], int length)
{
  int i;
  f[0] = 1;
  f[1] = 1;
  for(i = 2; i < length; i++)
  {
    f[i] = f[i-1] + f[i-2];
  }
  return length;
}


int main(int argc, char* argv[])
{
  int n = 4;
  int arr[] = {2, 4, 6, 8, 10, 12, 14, 16};
  int i;

  n = foo(arr, sizeof(arr)/sizeof(int));

  printf("n = %d\n", n);

  for(i = n/2; i < n; i++)
  {
    printf("arr[%d] = %d\n", i, arr[i]);
  }
}
```

8. The following program compiles and runs. What is its output? (10)

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

struct LNode
{
  int data;
  struct LNode* next;
};

struct LNode* createNode(int data)
{
  struct LNode* node = malloc(sizeof(struct LNode));
  node->data = data;
  node->next = NULL;
}

void printList(struct LNode* head)
{
  struct LNode* current = head;
  while(current != NULL)
  {
    printf("%d ", current->data);
    current = current->next;
  }
  printf("\n");
}

int main(int argc, char** argv)
{
  struct LNode* head = NULL;
  struct LNode* node = NULL;
  int i;
  for(i = 2; i < 13; i+=2)
  {
    node = createNode(i);
    node->next = head;
    head = node;
  }
  printList(head);
  node = head;
  while(node != NULL)
  {
    if(node->next != NULL)
    {
      node->data++;
      node->next = node->next->next;
    }
    node = node->next;
  }
  printList(head);
}
```

Additional scratch space for question 8



9. Using the Huffman coding algorithm, compute the codes for each character in the following string.

   `BANDANA`

   As we did on the assignment, if two trees in your priority queue have the same frequency, break the tie by making the tree with the larger symbol in the leftmost have the higher priority.

   (a) The code for A is: (2)

   (a) _____

   (b) The code for B is: (2)

   (b) _____

   (c) The code for D is: (2)

   (c) _____

   (d) The code for N is: (2)

   (d) _____