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.

1. (a) (2 points) Show how to use automatic variables to allocate room for an array capable of holding 40 integers on the stack.

   (b) (2 points) Show how to initialize all elements of above created array to 0. This has to be done separate from the array declaration.

   (c) (2 points) Show how to make the same allocation as in (a) but using dynamic memory allocation instead.

2. (2 points) A variable of type `void *` can be very useful. Why?

3. (2 points) C programming is said to be *close to the machine*. One remnant of assembly programming is that C contains a `goto` keyword that actully works. Why is the `goto` instruction considered by many hazardous to use?

4. (10 points) What is the output of this program?

```
1  #include <stdio.h>
2
3  void main(void)
4  {
5    unsigned char x = 21;
6
7    unsigned char a = x << 2;
8    unsigned char b = x >> 2;
9    unsigned char c = x & 13;
10   unsigned char d = x | 13;
11   unsigned char e = x ^ 13;
12
13   printf("a=%d, b=%d, c=%d, d=%d, e=%d\n",
14          a, b, c, d, e);
15 }
```

5. (12 points) What is the output of this program?

```
1  #include <stdio.h>
2
3  void main(void)
4  {
5    char s[] = "fQiQQnQalQ";
6    char del = 'Q';
7
8    int sourceIndex = 0;
9    int sinkIndex = 0;
10   while (s[sourceIndex])
11   {
12     if (s[sourceIndex] != del)
13     {
14       s[sinkIndex] = s[sourceIndex];
15       sinkIndex++;
16     }
17     else
18     {
19       printf("[%d,%d] %s\n", sourceIndex, sinkIndex, s);
20     }
21     sourceIndex++;
22   }
23   s[sinkIndex]='\0';
24   printf("result: %s\n",s);
25 }
```

6. (10 points) What is the output of this program?

```
1  #include <stdio.h>
2  #include <string.h>
3
4  char *findSubstring(char *str, char *target)
5  {
6    int len = strlen(target);
7    int n = 0;
8    while (*str)
9    {
10     printf("%c%c ",*str, *(target+n));
11     if ( *(target+n) == *str)
12     {
13       n++;
14       if (n == len) return (str-len)+1;
15     }
16     else
17     {
18       str -= n;
19       n = 0;
20     }
21     str++;
22   }
23   return NULL;
24 }
25
26 void main(void)
27 {
28   findSubstring("ABCDCDEF", "CDE");
29 }
```

7. (10 points) What is the output of this program?

```
1  #include <stdio.h>
2
3  int binarySearch(int x, int v[], int length)
4  {
5    int low, high, mid;
6    low = 0;
7    high = length-1;
8
9    while (low <=high)
10   {
11     mid = (low+high)/2;
12     printf("[%d %d %d] ", low, mid, high);
13
14     if (x < v[mid]) high = mid-1;
15     else if (x > v[mid]) low = mid+1;
16     else return mid;
17   }
18   return -1;
19 }
20
21 void main(void)
22 {
23   int nums[] = {12, 13, 15, 17, 21, 23, 27, 39, 43, 51};
24   printf("index = %d\n", binarySearch(17, nums, 10));
25   printf("index = %d\n", binarySearch(64, nums, 10));
26 }
```

8. (4 points) Consider the following code.

```
1 void main(void)
2 {
3   int a[] = {22,33,44};
4   int *x = a;
5   printf("sizeof(int)=%lu ", sizeof(int));
6   printf("x=%p, x[0]=%d\n", x, x[0]);
7   x = x + 2;
8   printf("x=%p, x[0]=%d\n", x, x[0]);
9 }
```

If the output from lines 5 and 6 is
`sizeof(int)=4 x=0x7fff29af6530, x[0]=22`
what is the output from line 8?

9. (5 points) The following program is compiled and run with the command: `./a.out 010123` What is the output?

```
#include <stdio.h>

int main(int argc, char* argv[])
{
  char* c_pt;
  int n = 0;
  if(argc == 2)
  {
    c_pt = argv[1];
    while(*c_pt)
    {
      if(*c_pt < '0' || *c_pt > '1') break;
      n = n*2 + *c_pt - '0';
      c_pt++;
    }
    printf("%d\n", n);
  }
}
```

10. (4 points) What is the output of this program?

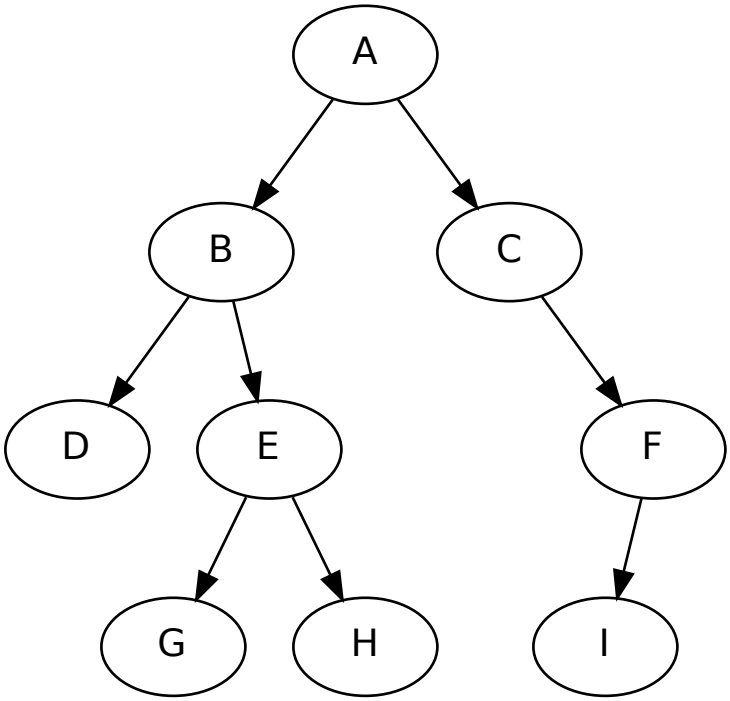
```
#include <stdio.h>

void main(void)
{
  char data[] = "hello";
  data[4] = '!';
  char *linePt = &data[3];
  *linePt = 'p';
  printf("[%s], [%s]\n", data, linePt);
}
```

11. (6 points) What is the output of this program?

```
1  #include <stdio.h>
2
3  struct Point
4  {
5    int x;
6    int y;
7  };
8
9  struct Point incPoints(struct Point p1, struct Point *p2)
10 {
11   p1.x++;
12   p1.y++;
13   p2->x++;
14   p2->y++;
15   return p1;
16 }
17
18 void main(void)
19 {
20   struct Point a = {1, 2};
21   struct Point b = {3, 4};
22   struct Point c = incPoints(a, &b);
23   printf("a=(%d, %d), b=(%d,%d), c=(%d,%d)\n",
24          a.x, a.y, b.x, b.y, c.x, c.y);
25 }
```

12. For the given tree, write out the following traversals.



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

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

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

13. (10 points) It's always intersting to see how different programming languages handle various operations. In Python and Matlab there a notion of array "slices". This makes it possible to do the following (Python):

```
>>> arr = range ( 0, 10 )
>>> print arr
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print arr[1:7:2]
[1, 3, 5]
```

In the above example there's a 10 element array, and the slice operator takes elements from start index 1 (inclusive), to end index 7 (exclusive), stepping by increment 2. Please show how to write a function called *slice* that does this in C. Your function should operate on an integer array and return a newly allocated array of the correct size. You do *not* need to perform boundary checks on the passed parameters, but your function should work if called with valid data.