Ananda Gunawardena

Course Review

Main Concepts

1. C data types, enum
2. printf, scanf
3. stdio.h, string.h, stdlib.h
4. preprocessing, macros
5. functions vs macros
6. extern, static, auto and register variables
7. pointers, *, **, ***
8. pointers and arrays, * vs [], ** vs [ ][ ]
9. malloc, calloc and free
10. makefiles, gdb
11. passing variables into functions, addresses and copies
12. const int* vs int* const
13. bit operations, masking bits, getbit and setbit
14. structs, typedefs
15. function pointers, void*
16. qsort, writing a comparator

Debugging Problems

You may be asked to find potential problems in a piece of code. Code may work correctly or may run into one of the common runtime errors. Some of the errors that you may encounter are

- Dereference of invalid or uninitialized pointer
- Insufficient or unallocated memory for the operation
- Reuse of storage after freeing it
- Double freeing of memory
- Incorrect use of pointer arithmetic
- Returning a pointer that is no longer valid
- Trying to free variables in the stack
- assigning incompatible types
- Array index out of bounds
- Program logic confuses pointer and reference types

In each of the following problems EITHER

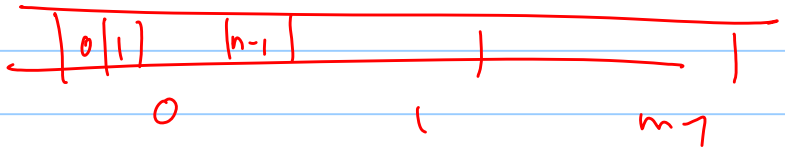
a) Find the correct output of the code

OR

b) The first line number where the program is likely to **crash, die, or express an incorrect result**

- The single best description of the underlying, root cause of the problem. (Not a description of the defective behavior that is exhibited)

int A[m][n]

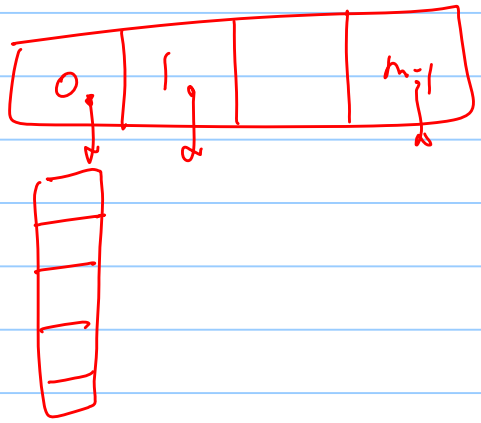


int** ptr;

ptr = malloc(m * sizeof(int*))

for(i=0; i<m; i++)

ptr[i] = malloc(n*sizeof(int))



ptr[0][1] = *(ptr[0]+1)

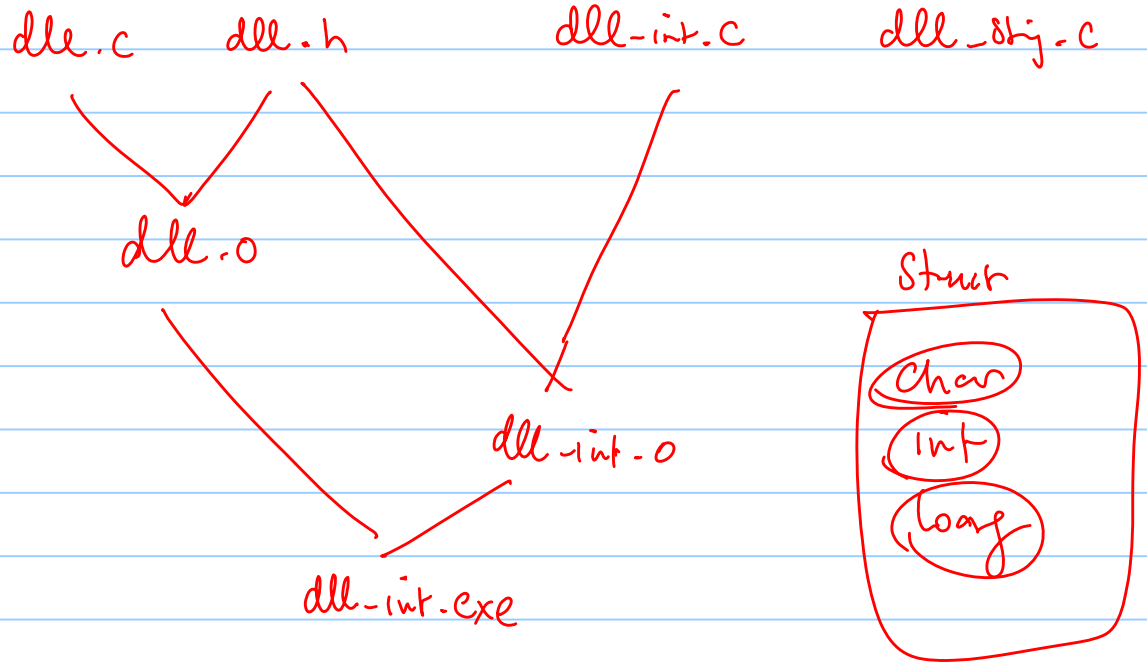
= *(*ptr+1)

---

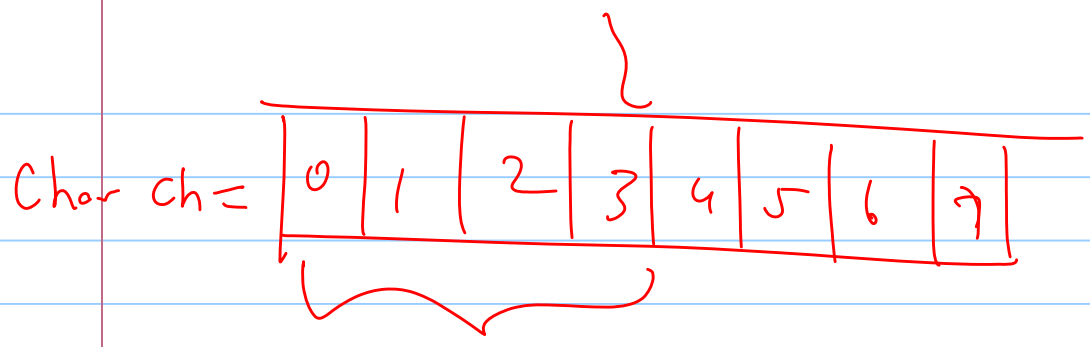
ptr = malloc(sizeof(name));

Char name[50];

Char* name;



int Comp(Const void* A, Const void* B) {



printf("%x", (ch>>4) & 0X0F);

ch = ch & 0x0F

ch = ch | 0XF0

Please note: As memory leaks do not directly cause a program to crash or produce erroneous output, they are not a listed among the potential answers for this portion of this exam.

Typical Format of the Question

```
1.      void Question1() {
2.        int* x, *ptr;
3.        *x=30;
4.        ptr=x;
5.        ptr++;
6.        printf("x=%d y=%d\n", *x, *ptr);
7.      }
8
9.      int main( int argc, char *argv[]) {
10.       Question1();
11.       return 0;
12.     }
```

==========

```
1.      void Question2() {
2.        int x, *ptr = &x;
3.        x=30;
4.        free(ptr);
6.        printf("x=%d y=%d\n", *x, *ptr);
7.      }
8
9.      int main( int argc, char *argv[]) {
10.       Question2();
11.       return 0;
12.     }
```

==========

```
1.      void Question3() {
2.        int* x, *ptr;
3.        ptr=malloc(30);
4.        *ptr=10;
5.        free(ptr++);
6.        printf("x=%d y=%d\n", *x, *ptr);
7.      }
8
9.      int main( int argc, char *argv[]) {
10.       Question3();
11.       return 0;
12.     }
```

==========

```
1.      void Question4(int* ptr) {
2.        int x;
3.        x=30;
4.        ptr=&x;
7.      }
8
```

```
9.      int main( int argc, char *argv[]) {
          int* ptr;
          Question4(ptr);
10.       printf("%d ", *ptr);
11.       return 0;
12.     }
```

==========

```
.       void Question5() {
2.        int* x, *ptr;
3.        ptr=malloc(8);
4.        int i;
5.        for (i=0; i<8; i++) *ptr+i = i;
6.        printf("x=%d y=%d\n", *x, *ptr);
7.      }
8
9.      int main( int argc, char *argv[]) {
10.       Question5();
11.       return 0;
12.     }
```

In each of the following questions, read the code and state the output or an incorrect approach.

```
[6] int main(){
  int* ptr = (int*)malloc(10*sizeof(int));
  int i;
  for (i=0;i<10;i++)
    *(ptr+i) = i;

  int** ptr2;
  for (i=0;i<10;i++)
      ptr2[i] = ptr+i;
  for (i=0;i<10;i++)
      printf("%d \n", *ptr2[i]);

  return 0;
}

[7]

#define n 10
int main(int argc, char* argv[]){
   int* A[n];
   doubleArray(&A,n);
   return EXIT_SUCCESS;
}

void doubleArray(int*** array, int n){
  int** arrayint = (int**)malloc(2*n*sizeof(int*));
  int i;
  for (i=0;i<n;i++)
    arrayint[i] = (*array)[i];
  free(*array);
  array = &arrayint;
```

```
}
```

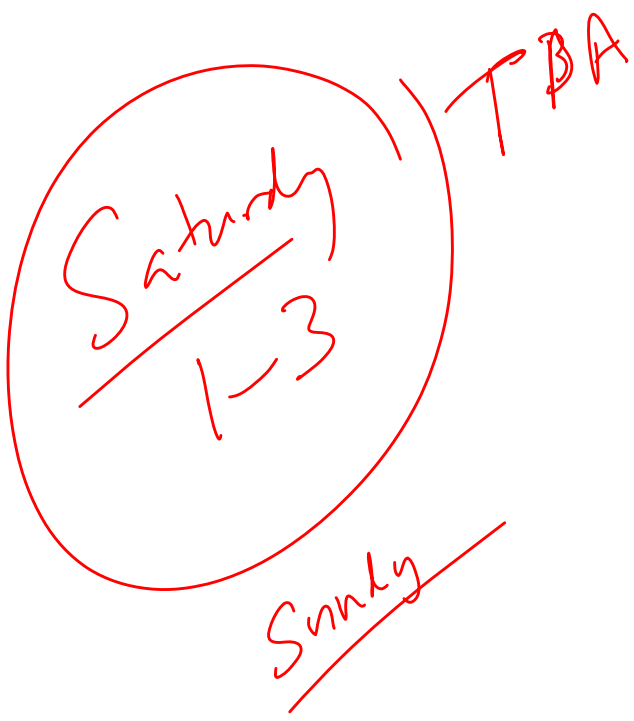
```
[8] int main(int argc, char* argv[]){
   char** ptr = malloc(10*sizeof(char*));
   ptr[0] = malloc(strlen("guna")+1);
   strcpy(ptr[0],"guna\0");

   free(ptr);
   printf("%s \n", ptr[0]);
   free(ptr);
   printf("The address of ptr is %x ptr[0] is %x \n", ptr, ptr[0]);
   return EXIT_SUCCESS;
}
```

```
[9] int main(){
  int* ptr = (int*)malloc(10*sizeof(int));
  int i;
  for (i=0;i<10;i++)
    *(ptr+i) = i;

  ptr += 5;
  free(ptr);
  return 0;
}
```

```
[10]
   char** ptr = malloc(10*sizeof(char*));
   for (i=0;i<10;i++)
      ptr[i]=malloc(i);

   free(ptr);
   printf("%s \n", ptr[0]);
   free(ptr[0]);
```

```
[11] int main(){
  int* ptr = (int*)malloc(sizeof(int));
  int i = 12;
  ptr = &i;
  free(ptr);
  return 0;
}
```

```
[12] int main(){
  int n = 10,i;
  void* A = malloc(n*sizeof(int));
  for (i=0;i<=n;i++)
    *(A+i) = i;
  return 0;
}
```




```
[13] int main() {
     char* name = malloc(sizeof(char*));
     strcpy(name, “gunawardena”);
```

```
    printf("%10s \n", name);
}
```

Declare a variable: `$myname = 'guna';`

- **Declare an array:** `@A = (guna, me, him, her);`
- **Declare a hash table:**
  `%table = {'key1','value1','key2,'value2'}`
- **Print a variable:** `print $myname;`
- **Open a file:** `open(INFILE, "filename");`
- **Print to a file:** `open(out, ">filename"); print out "something";`
- **Read a file:** `foreach $line (<INFILE>) { do_something }`
- **remove the newline character:** `chomp($line);`
- **Split a String:** `@array = split(/regex/, $line)`
- **Find the Sum:**
  `$sum = 0; for ($i=0 ; $i<$size; $i++) { $sum += $i; }`
- **Check if the file name is "guna":**
  `if ($file eq 'guna') { do_something }`
- **Open a dir and read files:**
  `opendir(DIR,"."); foreach $file (readdir(DIR) ){ .. }`
- **Make a directory:** `mkdir $dirname;`
- **Remove a directory:** `rmdir $dirname;`
- **copy a file:** `use File::Copy; copy($file1, $file2);`
- **rename a file:** `rename($file1, $file2);`
- **using regex:** `while ($line =~ /regex/g ) { print $1;}`
- **Command Line Arguments:** `$#ARGV` - index of last argument, `$ARGV[0]` - first arg
- **Running a perl script from another:** `system 'perl program.pl text.txt'`

1. Write a perl script that runs as follows
   a. perl script.pl input.txt searchword
   look for searchword in input.txt file

2. Write a perl script that looks for a pattern in a string S, the pattern starts with a digit and has even number of the pattern "ab" (consecutive) with any other characters in between
3. Write a shell script that takes a folder and list all directories in the folder
4. What are shell variables, environment, path variables, unix shell, processes, ps, pid, background, foreground processes