601.220 Intermediate Programming
MIDTERM REVIEW QUESTIONS

These sample questions are meant to roughly convey the question formats and topic coverage of the midterm. This is not a sample midterm, in that it is not meant to be representative of the actual length of the midterm.

Choose the SINGLE BEST option for each of the following questions. Clearly indicate your choice; ambiguous answers will be marked incorrect. Assume appropriate variable declarations have been made and the necessary header files have been included.

---

1) Which of the following declares a variable 'dptr' which will be a pointer to a double number?

```
a) double & dptr;                b) dnum * dptr;

c) double * dptr;                d) dptr -> double;
```

2) The following code segment uses an `int *` variable called `arr`:

```
int n = 0;
while(*arr < 0) {
    arr++;
    n++;
}
printf("%d\n", n);
```

Which of the following declarations for `arr` is NOT compatible with this code segment?

```
a) int arr[];                    b) const int *arr;

c) int * const arr;              d) int *arr;
```

3) Let a and b be integer variables and let a=1, b=4. Which expression DOES NOT evaluate to true?

```
a) a / b                         b) a - b

c) a % b                         d) b - a
```

4) Consider this code segment:

```
int my_func() {
    int a = 4;
    for(int i = 0; i < 10; i++) {
        // LOOKY HERE
        for(int j = 0; j < i; j++) {
            int c = i * j;
            a += c;
        }
    }
    return a;
}
```

Which variables are in scope at the point indicated by "LOOKY HERE"?

a) a and i

b) a, i and c

c) a, i, j and c

d) only c

5) Which of the following prints "true"?

a) `if(0) { printf("true\n"); }`

b) `if(0 && 1) { printf("true\n"); }`

c) `if(-1) { printf("true\n"); }`

d) `if(!4) { printf("true\n"); }`

6) Which of the following prints "true"?

a) `if(1 >> 1) { printf("true\n"); }`

b) `if(0 << 4) { printf("true\n"); }`

c) `if(1 & 0) { printf("true\n"); }`

d) `if(1 >> 0) { printf("true\n"); }`

7) Assume `int a` is greater than 0. Which of the following is true when a is even and false when a is odd?

a) `(a & 1) == 0`

b) `(a >> 1) != 0`

c) `(a / 2) == 0`

d) `(a % 2) != 0`

8) Let i be an integer variable initially equal to 0. Which of the following prints a *FINITE* and *NON-EMPTY* list of *EVEN* numbers?

a)
```
while(i > 0 && i < 10) {
    printf("%d ", i);
    i += 2;
}
```

b)
```
while(i > 0 || i < 10) {
    printf("%d ", i);
    i += 2;
}
```

c)
```
do {
    printf("%d ", i);
    i += 2;
} while(i > 0 && i < 10);
```

d)
```
do {
    printf("%d ", i);
    i += 2;
} while(i > 0 || i < 10);
```

9) Which of the following FAILS to print all the characters in the zero-terminated string s ?

a)
```
char *cur = s;
while(*cur != '\0') {
    printf("%c", *cur);
    cur++;
}
```

b)
```
char *cur = s;
while(*cur != '\0') {
    printf("%c", cur[0]);
    cur++;
}
```

c)
```
int i = 0;
while(i < strlen(s)) {
    printf("%c", s[i++]);
}
```

d)
```
for(int i = 0; i < strlen(s); i++) {
    printf("%c", s[i++]);
}
```

10) Which of the following assigns the address of variable 'char ch' to variable 'char *cptr' ?

a) `&ch -> cptr;`

b) `cptr = ch;`

c) `*cptr = *ch;`

d) `cptr = &ch;`

11) Which of the following assigns value '?' to variable 'char ch' assuming that 'char *cptr' points to 'ch' ?

a) `&cptr = '?';`

b) `cptr = '?';`

c) `*cptr = '?';`

d) `cptr = & '?';`

12) Consider the following struct:

```
struct some_data {
    char *name;
    char student_id[10];
    float *grade_list;
};
```

If sizeof(char) == 1 and sizeof(void *) == 8 and the struct is not padded, what is sizeof(struct some_data)?

a) 36

b) 96

c) 28

d) 26

13) Consider this code segment:

```
int a = 5;
float b = a / 2;
```

Which of the following statements is true?

a) At the end, b equals 2.0

b) At the end, a's type has changed to float

c) The a operand's type is promoted to float prior to the division

d) The 2 operand's type is promoted to float prior to the division

14) Which of the following declares a zero-terminated string initialized to "work it" ?

a) `char str[] = 'work it';`

b) `char str[7] = "work it";`

c) `string str = "work it";`

d) `char str[8] = "work it";`

15) Which of the following prints the character 'i' using 'str' from the previous question?

a) `printf("%p", str+5);`

b) `printf("%c", str+6);`

c) `printf("%c", *(str+5));`

d) `printf("%c", *(str+6));`

16) Which of the following reads a string from standard input into character array 'word'?

a) `scanf("%c", word);`

b) `scanf("%s", &word);`

c) `scanf("%s", word);`

d) `scanf("%s", *word);`

17) Say 's1' and 's2' have type 'char *', with 's2' pointing to a string and 's1' pointing to an uninitialized array of chars. Which code snippet below successfully copies the contents of string 's2' into 's1', including the terminator? Assume 's1' has enough space.

a)
```
s1 = s2;
```

b)
```
while(*s2 != '\0') {
    *s1++ = *s2++;
}
```

c)
```
do {
    s1[i] = s2[i];
    i++;
} while(s2[i-1] != '\0');
```

d)
```
for(int i = 0; i < strlen(s2); i++) {
    s1[i] = s2[i];
}
```

18) Which is the correct prototype for the `calloc` function?

a) `void calloc(size_t)`

b) `void calloc(size_t, size_t)`

c) `void *calloc(size_t)`

d) `void *calloc(size_t, size_t)`

19) A memory leak occurs when:

a) a locally declared array goes out of scope

b) two functions make modifications to the same address in memory

c) dynamically-allocated memory is not freed before the program exits

d) a function returns a pointer to a locally-declared array

20) You are adding elements to a dynamically-allocated array and you eventually discover that the array is full. Which function is most appropriate for making room for the new elements?

a) strlen b) realloc

c) scanf d) free

21) Which of the following declares a structure containing a character array with enough space to hold a 6-letter word as a C-style string, as well as two real numbers?

a) `struct s5 { char car[7]; float n1; float n2; };`

b) `struct s5 { char car[6]; float n1; float n2; };`

c) `struct s5 { char car[7]; int n1; int n2; };`

d) `struct s5 { char car[6]; int n1; int n2; };`

22) Which of the following declares a structure containing an integer, a character and a field of the structure type in the above question?

a) `struct s6 { int num, char ch, s5; };`

b) `struct s6 { int num; char ch; s5 s5var; };`

c) `struct s6 { int num; char ch; struct s5 s5var; };`

d) `struct s6 { int num, char ch, struct s5; };`

23) Which of the following declares a variable to be of the structure type from the previous question?

```
a) s6 s6var;                    b) typedef struct s6 s6var;

c) typedef s6 s6var;            d) struct s6 s6var;
```

24) Assuming 'var5' has been declared a variable of the structure type s5 above, which of the following initializes its character array to the empty string?

```
a) var5.car = "";               b) var5->car = "";

c) var5->car = '\0';            d) var5.car[0] = '\0';
```

The next few questions assume the following definitions:

```
struct it {
    int num;
    char word[5];
};

typedef struct thing {
    float f;
    struct it t;
} TYPEthing;

struct it s2 = {3, "yo"};

TYPEthing t1;
```

25) Which of the following initializes t1 so that all numbers are 0 and all strings are empty?

```
a) t1.f = 0.0f; t1.num = 0; t1.word[0] = '\0';

b) t1.f = 0.0f; t1.t.num = 0; t1.t.word[0] = '\0';

c) t1.f = 0.0f; t1.t->num = 0; t1.t->word[0] = '\0';

d) t1->f = 0.0f; t1->t->num = 0; t1->t->word[0] = '\0';
```

26) Which of the following reads the next word in the text file pointed to by 'fptr' into the word field of variable s2 above?

```
a) scanf(fptr, "%s", s2.word);     b) fscanf(fptr, "%s", s2.word);
```

```
c) fscanf(fptr, "%s", &s2.word);   d) scanf("%s", s2.word);
```

27) Which of the following sets file pointer 'fptr' for reading the existing file 'f1.txt' ?

```
a) fptr = fopen("f1.txt", "r");     b) fopen(fptr,"f1.txt", r);

c) fptr = fopen('f1.txt', 'r');     d) fopen(fptr,"f1.txt", e);
```

Short-answer questions. Please respond as indicated. When writing C code, include all required punctuation and aim for correct syntax. Use comments to explain your approach.

---

28) What is the relative order of the following four operations when creating and running a C program? (number them 1-4)

```
____  edit

____  execute

____  preprocess

____  compile
```

29) Write the function prototype (declaration) for a function called 'thing' that has two float parameters and returns nothing.

30) strcpy is a function that copies the characters in a string (second argument) to a destination array (first argument), including the null terminator. In the blank space below, write a new version of the first line of this code segment such that the strcpy succeeds. Use only stack memory and try not to waste memory.

```
char *s1;       // change this line
char *s2 = "good luck";
strcpy(s1, s2); // I must succeed
```

31) Why does the call to realloc in the following code segment fail?

```
int ra[20];
ra = realloc(ra, 10 * sizeof(int));
```

Write-a-function questions. Please respond as indicated. When writing C code, include all required punctuation and aim for correct syntax. Use comments to explain your approach.

---

32) Write a C function called 'descending' that takes two integer parameters and returns a dynamically-allocated array of consecutive integers. The first parameter specifies the number of elements in the array and the second specifies the first number to appear in the array. The array elements should be descending, so 'descending(3, 7)' should return a dynamically allocated array containing {7, 6, 5}.

33) Write C statements which use the above function to create a descending array with 10 elements where the first number in the list is -20. Assign the result to a variable. Don't worry about freeing memory.

34) Write a C function called `shifty` that takes a string (character array) and an integer shift value as parameters. The function should change the string such that each alphabet character in the original string is replaced by a new character which is shifted forward in the alphabet by the specified shift value, where the value ranges from -25 to +25. When the end of the alphabet is reached, wrap around to the beginning when shifting. Preserve capitalization. For example,

```
char s[] = "Good luck! 2u";
shifty(s, 16);        // no value is returned by this function
printf("%s\n", s);     // results in "Weet bksa! 2k"
```

because `abcdefghijklmnopqrstuvwxyz` becomes `qrstuvwxyzabcdefghijklmnop` when the call `shifty(s, 16)` is made. You may use functions from the ctype library.

Indicate the output of the given program or code segment.

---

35) What is the output of the following program?

```
#include <stdio.h>

int fa (int a, int *b) {
   int c = 5;
   a = 20;
   *b = a / 2;
   return *b / 2;
}

int main() {
   int c = 6, b = 12, d = fa(3, &b);
   printf("1) %d %d %d\n", d, c, b);

   c = 6; b = 12; d = fa(b, &c);
```

```
        printf("2) %d %d %d\n", d, c, b);

        return 0;
    }
```

>>> OUTPUT:

1)

2)

36) What is the output of the following program? (Ignore any memory leaks.)

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

  struct node {
      char ch;
      struct node * next;
  };

  // Print each list node's 'ch' character, in order
  void pout(struct node *head) {
      while(head != NULL) {
          printf("%c ", head->ch);
          head = head->next;
      }
      printf("\n");
  }

  struct node *mystery(struct node *head) {
      struct node *cur = head;
      while(cur != NULL) {
          struct node *temp = malloc(sizeof(struct node));
          temp->ch = cur->ch;
          temp->next = head;
          head = temp;
          cur = cur->next;
      }
      return head;
  }



 //code continued on next page
```

```
    int main() {
        struct node *head;
        struct node *temp;
        char ch = 'A';
        head = malloc(sizeof(struct node));
        head->ch = ch;
        head->next = NULL;
        temp = head;
        for(int i = 1; i <= 2; i++) {
            temp->next = malloc(sizeof(struct node));
            temp = temp->next;
            temp->ch = ch + i;
            temp->next = NULL;
        }
        printf("a) ");
        pout(head);
        printf("\nb) ");
        pout(mystery(head));
        return 0;
    }
  /*********************************************************/
```

>>> OUTPUT:

a)

b)

37) What is the output of the following program?

```
#include <stdio.h>
void display();

int main() {
    display();
    display();
}
void display() {
    static int c = 0;
    printf("%d  ",c);
    c += 5;
}
```

>>> OUTPUT:

38) Does the following code compile without errors/warnings? If so, what is the output? If not, why not?

```
#include <stdio.h>

int k = 20;

void fun(int k) {
  printf("%d ", k);
}

int main() {
  printf( "%d ", k);
  int k = 10;
  printf( "%d ", k);
  {
    int k = 0;
    k++;
    printf( "%d ", k);
  }
  printf( "%d ", k);
  k++;
  fun(k);
  return 0;
}
```

>>> OUTPUT:

39) Which of the following successfully creates a ***3 * 5 2D*** *array on the heap* (i.e., using dynamic memory allocation)? Note that 3 * 5 means a 2D array that has 3 rows and 5 columns.

a.
```
int **a = malloc(sizeof(int *) * 3);
for (int i = 0; i < 3; i++) {
      a[i] = malloc(sizeof(int) * 5);
}
```

b.
```
int **a = malloc(sizeof(int *) * 5);
for (int i = 0; i < 5; i++) {
      a[i] = malloc(sizeof(int) * 3);
}
```

c.
```
int *a = malloc(sizeof(int) * 15);
```

d.
```
int *a = malloc(sizeof(int) * 3);
 for (int i = 0; i < 3; i++) {
      a[i] = malloc(sizeof(int) * 5);
 }
```

40) Assume the following is a function that calculates and returns the square root of non-imaginary numbers. Note that we have replaced the actual code that does the calculation with a comment line. Which assert statement would be most sensible and comprehensive to be the first line in the body of this function?

```
double sqrt (double val) {
      // WHICH ASSERT STATEMENT SHOULD GO HERE?
      // code to calculate and return the square root of val
}
```

a) `assert( val > 0 );`
b) `assert( val != 0 );`
c) `assert( !(val <= 0) );`
d) `assert( !(val < 0) );`

41) Write code to properly read at once all the 100 double values that are stored (in binary format) in a file named “doubs.dat” and store them in ray_doub. You do NOT need to do any error checking and/or make sure that you successfully read those 100 values.

```
#include <stdio.h>

int main() {
      double ray_doub[100];
      FILE *fp = fopen("doubs.dat", "rb");
      // WRITE CODE HERE …
      fclose(fp);
}
```