Quiz 2 Review

c/c++205 summer

1. Which data structure should I use if I want to store the marks of each student in a course in a sorted order?

a. use an array because it's inexpensive to insert/delete elements in an array compared to a linked list

b. use a linked list because it's inexpensive to insert/delete elements in a linked list compared to an array

c. use arrays because the number of students differs each year

d. use linked lists because the number of students differs each year

e. both answers that recommend arrays are correct

f. both answers that recommend linked lists are correct

2. How will you allocate an array which can store a maximum of 150 integers?

a. int *str = malloc(150);

b. int *str = free(150);

c. int *str = (int*) malloc(150*sizeof(int));

d. Both the first and third answers

3. The following code snippet:

```
char s[10] = "Hello";
char *p = s;
int i;
for (i=0; i < strlen(s); i++) {
  putchar(toupper((*p)++));
}
putchar('\n');
```

would display:

a. HELLO

b. HIJKL

c. Hello

d. Nothing, it crashes

4. The following code snippet:

```
char  string[15] = "Shenzhen";
char *s;
int   i;

s = string + 3;
for (i = 0; i < 3; i++) {
  putchar(*(s++));
}
putchar('\n');
```

displays:

a. qzh

b. nzh

c. She

d. Vhe

e. Vkh

5.

```
 typedef struct my_node {
     char* info;
     struct my_node *next;
} NODE_T;

int main() {
  NODE_T *p = NULL;
  p = (NODE_T*)malloc(sizeof(NODE_T));
  return 0;
}
```

What is p?

a. A variable of type NODE_T

b. A string of type NODE_T

c. A pointer of type NODE_T

d. A pointer to a block of memory of size, sizeof(NODE_T)

e. Both C and D

6. If a variable is a pointer to a structure, then which of the following operators is used to access data members of the structure through the pointer variable?

a. -> (arrow)
b. . (dot)
c. * (star)
d. & (ampersand)

7. What will be the output of the following program?

```
#include <stdio.h>

typedef struct course {
        int  courseno;
        char coursename[25];
    } COURSE_T;

int main() {
    COURSE_T c[] = {{205, "C/C++"},
                    {307, "Databases"},
                    {209, "Java"}};

    printf("%d ", c[1].courseno);
    printf("%s\n", (*(c+2)).coursename);
    return 0;
}
```

a. 205 Databases
b. 307 Databases
c. 307 Java
d. 205 C/C++

8. What will be the output of this program?

```
#include <stdio.h>
int func(int n) {
    int s, d;

    if (n!=0) {
      d = n%10;
      n = n/10;
      s = d + func(n);
    } else {
      return 0;
    }
    return s;
}

int main() {
    int a = func(123);
    printf("%d\n", a);
    return 0;
}
```

a. 4 b. 3 c. 6 d. 12

9. If we want to correctly free memory in the program, we must add:

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

typedef struct {
        int   i;
        float j;
        char *s;
    } struct_t;

int main() {
    struct_t *p;
    p = (struct_t *)malloc(sizeof(struct_t));
    p->s = (char*)malloc(20);
    return 0;
}
```

a. free(p); free(p->s);
b. free(p->s); free(p);
c. free(p->s);
d. free(p);

10. Consider the code snippet below.

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

int main() {
        int *p;
        int *q;

        p = (int *)malloc(256);
        p = (int *)malloc(512);
        q = realloc(p, 1024);
        printf("How much memory is reserved?\n");
        return 0;
}
```

When the question is printed to the screen, the amount of heap memory allocated to the program at this very moment is:

a. 256 bytes
b. 512 bytes
c. 1024 bytes
d. 1280 bytes (1024 + 256)
e. 1536 bytes (1024 + 512)
f. 1792 bytes (1024 + 512 + 256)