Practice Test 2, Program Design

This practice test has more questions than a regular midterm. It has 13 short-answer questions and 7 free-response questions. You can expect 10 short-answer questions and 4 free-response questions in the midterm.

1. Suppose the following declarations are in effect:

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &a[1], *q = &a[5];
```

What is the value of

1) `*(q-3)`
2) `p - q`
3) `*p - *q`?

Answer:

2. What is the value of sum after the following program fragment is executed?

```
#define N 5
int a[N] = {6,2,3,9,4};
int sum=0, *p;
for(p=a; p < a+N; p++)
    if((p-a)%2 == 1)
        sum += *p;
```

Answer:

3. What will be the contents of `a` array after the following statements are executed?

```
#define N 10
int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *p = a, *q = a+N-1;
while(p<q){
   *p = *q;
   *q= *p;
   p++;
   q--;
}
```

Answer:

4. What is the output of the following program fragment?

```
#define N 5
int a[N] = {4,1,5,6,3};
int *p=a, *q=a+N-1;
while(p < q) {
    if((*p+*q)%2 == 1) {
        p++;
    } else {
        q--;
    }
}
printf("%d\n", *q);
```

Answer:

5. What is the output of the following program?

```
#include <stdio.h>
void func(int *a, int n);
int main()
{
  int a1[5]={0};
  func(a1, 5);
  printf("%d", a1[3]);
  return 0;
}

void func(int *a, int n)
{
  int *p;
  *a=0;
  for (p=a+1; p<a+n; p++)
    *p = p - a + *(p -1);
}
```

Answer:

6. An array of strings is defined as:

```
char *planets[] = {"Mercury", "Venus", "Earth", "Mars", "Jupiter",
"Saturn", "Uranus", "Neptune"};
```

What's the value of `planets[1][3]`?

Answer:

7. Let f be the following function:

```
int f(char *s, char *t)
{
    char *p1, *p2;
    for(p1 = s; *p1 != '\0'; p1++){
              for(p2 = t; *p2 != '\0'; p2++)
                if (*p1 == *p2) break;
         if(*p2 == '\0') break;
    }
    return p1 - s;
}
```

What is the return value of f("cabd", "acad")?

Answer:

8. Consider the following program max_min.c:

```
 1 #include <stdio.h>
 2
 3 void max_min(int *a, int n, int *max, int *min);
 4
 5 int main(void)
 6 {
 7         int a[]={6, 8, 14, 5, 9, 23, 45, 65};
 8
 9         int max_a, min_a;
10
11         max_min(a, 8, &max_a, &min_a);
```

```
12          printf("max is %d and min is %d\n", max_a, min_a);
13          return 0;
14
15 }
16 void max_min(int *a, int n, int *max, int *min)
17 {
18   int *p;
19   max = min = *a;
20   for (p = a; p < a + n; p++) {
21     if (*p > *max)
22       *max = *p;
23     else if (*p < *min)
24       *min = *p;
25   }
26 }
```

This program compiles but when it runs, it causes a segmentation fault. Which line causes the segmentation fault?

Answer:

9. What will be the value of string s2 after the following statements have been executed?

```
strcpy(s1, "Program");
strcpy(s2, "Design");
if (strcmp(s1, s2) < 0)
      strcat (s1, s2);
else
      strcat(s2, s1);
```

Answer:

10. Which one of the following definition of the swap function would exchange the values of the variables i and j successfully? `swap(&i, &j);  /* exchange values of i and j */`

```
a)
void swap(int *x, int *y)
{       int *tmp;
         tmp = x;
         x=y;
         y=tmp;
}
b)
void swap(int *x, int *y)
{       int tmp;
        tmp = *x;
         *x=*y;
         *y=tmp;
 }
 c)
 void swap(int *x, int *y)
 {       int tmp;
        tmp = *x;
         *y=*x;
         *x=tmp;
}
```

Answer:

11. Which of the following program will echoes its command-line argument in reverse order? Running the program by type `./a.out today and tomorrow` should produce the output `tomorrow and today.`

a.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
 int i;
 for (i = argc; i >= 0; i--)
       printf("%s ", argv[i]);
 printf("\n");

 return 0;
}
```

b.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
 int i;
 for (i = argc; i > 0; i--)
       printf("%s ", argv[i]);
 printf("\n");

 return 0;
}
```

c.

```
#include <stdio.h>

int main(int argc, char *argv[])      {
 int i;
 for (i = argc - 1; i >= 0; i--)
       printf("%s ", argv[i]);
 printf("\n");

 return 0;
}
```

d.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
 int i;
 for (i = argc -1; i > 0; i--)
       printf("%s ", argv[i]);
 printf("\n");

 return 0;
}
```

Answer:

12. When a user provides command-line arguments that are the same in reverse order, we call them palindromic arguments. For instance, the command-line "./a.out 1 20 5 20 1" has palindromic arguments, but the command-line "./a.out 1 2 3 4 5" does not contain palindromic arguments. Which of the following programs correctly checks if its command-line arguments are palindromic? It should print "YES" if they are palindromic, and "NO" otherwise.

a.
```
#include <stdio.h>
int main(int argc, char *argv[]) {
  int i, flag=1;
  for(i=0; i < argc; i++) {
    if(argv[i] != argv[argc-i-1])
        flag=0;
  }
  if(flag)
    printf("YES\n");
  else
    printf("NO\n");
  return 0;
}
```

b.
```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
  int i, flag=1;
  for(i=0; i < argc; i++) {
    if(strcmp(argv[i], argv[argc-i-1]))
        flag=0;
  }
  if(flag)
    printf("YES\n");
  else
    printf("NO\n");
  return 0;
}
```

c.
```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
  int i, flag=1;
  for(i=1; i < argc; i++) {
    if(strcmp(argv[i], argv[argc-i]))
        flag=0;
  }
  if(flag)
    printf("YES\n");
  else
    printf("NO\n");
return 0;
}
```

d.

```
#include <stdio.h>
int main(int argc, char *argv[]) {
  int i, flag=1;
  for(i=1; i < argc; i++) {
    if(argv[i] != argv[argc-i-1])
        flag=0;
  }
  if(flag)
    printf("YES\n");
  else
    printf("NO\n");
  return 0;
}
```

Answer:

13. What does the following `read_line` function do, assuming that n is larger than the size of the input string?

```
int read_line(char *str, int n) {
          int ch, i = 0;
          while((ch = getchar()) == ' ') ;
          *str++= ch;
          i++;
          while ((ch = getchar()) != '\n') {
               if (i < n) {
                 *str++= ch;
                 i++;
               }
          }
          *str = '\0';
          return i;
}
```

a) reads in a line of input but skips white space before beginning to store input characters
b) reads in a line of input but skips white space and the first non-white-space character before beginning to store input characters
c) reads in a line of input but stops reading at the first white-space character
d) reads in a line of input and stop reading at the first new-line character but does not store white space characters in the string

Answer:

Free-From Questions

14. Complete the program below so it calls the swap function and prints the words in the array w in alphabetical order.

```
#include <stdio.h>
void swap(char *a, char *b) {
    char tmp[4];
    strcpy(tmp, a);
    strcpy(a, b);
    strcpy(b, tmp);
}
int main(){
  char w[3][4] = {"dog", "rat", "cat"};









  for(int i=0; i < 3; i++)
    printf("%s\n", w[i]);
}
```

15. Write the following function:

```
void split_time(int total_sec, int *hr, int *min, int *sec);
```

`total_sec` is a time represented as the number of seconds since midnight. `hr`, `min`, and `sec` are pointers to variables in which the function will store the equivalent time in hours (0 -23), minutes (0-59), and seconds (0-59), respectively.

16. Complete the following function that replaces every occurrence of the character passed in as the second parameter by the character passed in as the third parameter in the string *str* passed in as the first parameter. For example, a function call replace(sentence, 'r', 't') will replace every occurrence of 'r' with 't' in the string sentence.

```
void replace(char *str, char x, char y){




}
```

17. A binary bar code scan is a bit pattern that contains only 1s and 0s. Write a function that finds the edges of light and dark regions of a binary code. Process an input bit pattern in the following manner:
    - Assign a 1 to the output bit pattern whenever two consecutive bits (one bit and it's previous bit) are different
    - Assign a 0 to the output bit pattern whenever two consecutive bit (one bit and it's previous bit) are the same
    - Assign 0 to the first output bit since there is no previous bit for the first bit

    For example, input and output bit pattern of the program that detects the edges might look like the following:
    Input: 00101101
    Output: 00111011

Write the function edge() with the following prototype to perform edge detection. The arguments of the function `edge()` contain the length of the input and output arrays with the same size, the input array `a1`, and output array `a2`. **Use pointer arithmetic – not subscripting- to visit array elements. In other words, eliminate the loop index variables and all use of the [] operator in the functions.**

```
void edge(int *a1, int *a2, int n) {









}
```

18. Complete the function that extract the extension from a file name. `file_name` points to a string containing a file name. The function should store the extension on the file name in the string pointed to by `extension`. For example, if the file name is “memo.txt”, the function will store “txt” in the string pointed to by `extension`. If the file name doesn’t have an extension, the function should store an empty string (a single null character) in the string pointed to by `extension`. String library functions are not allowed.

```
void get_extension(char *file_name, char *extension) {
```

```
}
```

19. Complete the function is_all_uppercase so that it checks if all letters in an input string str are uppercase letters. The string str is composed of letters only. It must return 1 if all letters in str are uppercase, and 0 otherwise.

```
int is_all_uppercase(char str[]) {
  int flag = 1;









  return flag;
}
```

20. Write the function that shifts a message. The function expects `message` to point to a string containing the message to be shifted; `shift` represents the amount by which each letter in the message to be shifted. Lower-case letters remain lower-case when shifted, and upper-case remain upper-case. For example, if the `message` is `"Go ahead, make my day."`, and `shift` is `3`, the function will modify `message` to `"Jr dkhdg, pdnh pb gdb."` If the message is `"Jr dkhdg, pdnh pb gdb."`, and `shift` is `23`, the function will modify `message` to `"Go ahead, make my day."`. Hint: To handle the wrap-around problem, you could use the % operator. For example, `'X'` shifted by 4 is `'B'`.

```
void shift(char *message, int shift) {




}
```