1) **This function is part of a program that is running on a 32-bit x86 system; the compiler does not change the order of variables on the stack.**

```
void function(char *input) {
        int i = 1;
        char buffer[8];
        int j = 2;
        strcpy(buffer,input);
        printf("%x %x %s\n",i,j,buffer);
}
```

What is the minimum length of a string – passed to the function through the input parameter – that can crash the application?

A. 9
B. 10
C. 11
D. 12
E. 13

2) **Which of the following is true with respect to buffer overflows?**

A. Buffer overflows on the heap cannot be exploited to run arbitrary code.
B. If a function is vulnerable to a buffer overflow due to large user input being put in a small fixed-size buffer, making the buffer 10 times as large as a "quick fix" will reduce the impact of the vulnerability.
C. Buffer overflows can be used to alter the state and operation of the vulnerable application in an undetectable way.
D. If code cannot be executed on the stack (e.g. through the use of the non-execute bit or DEP), attackers cannot run arbitrary code by exploiting a buffer overflow.
E. Calling free() on the same memory address twice may crash the application, but will not lead to an exploitable buffer overflow.

3) Identify all the problems with this code

```
#include <string.h>

#define goodPass "GOODPASS"

int main()
{
   char passIsGood = 0;
   char buf[80];

   printf("Enter password ");
   gets(buf);

   int len = strlen(buf);
   if(len < strlen(goodPass)
      passIsGood = 0;

   if(strcmp(buf,goodPass)==0)
      passIsGood = 1;
   if(passIsGood == 1)
      printf("You Win\n");
}
```

4) How would you fix the code below?

```
#define ll 12

char pwd[37], n[ll];

void s(char *u){
   strncpy(n,u,ll);
   printf(n);
}
```

5) When dealing with Unicode user input in C, the following issues need to be taken into account:

A. Unicode characters may be used to bypass black-list filtering
B. In every encoding form, the size of Unicode characters may differ from each other
C. The length() of a Unicode string may be different from its size()
D. Unicode strings cannot be printed easily out on the screen.
E. Directional control characters such as U+202E may be exploited