Tobias Brunner
08-102-204

Esther Hoeun Yu

CA EX 1

Q1

Word[7] = [‘s’,’t’,’r’,’i’,’n’,’g’,’\0’]

Q2

```
// original with added checks for array out of bound
int setValueAt(double *x, int i, double value, int l){
    if (i <= l){
        x[i] = value;
        return 0;
    } else{
        return 1;
    }
}

// improved
int setValueAt(double *x, int i, double value, int l){
    if (i <= l){
        *(x+i) = value;
        return 0;
    } else{
        return 1;
    }
}
```

Tobias Brunner
08-102-204

Esther Hoeun Yu

Q3

```
long a = 1234567890; /* Hex: 499602d2 */
long b = 1000000000; /* Hex: 3b9aca00 */

// void pointer to b
void*p=&b;

// print address of b since p points to this address
printf("%x\n", p);

// pointer is cast to a pointer of type long and incremented after print
// prints b
printf("%x\n",*(long*)p++);

// pointer p and cast to a pointer of type char which is promoted to int in
the print statement
// and is incremented after print
// prints ffffffca
printf("%x\n",*(char*)p++);

//
// prints ca
printf("%x\n",*(unsigned char*)p++);

//
// prints e6222a03
printf("%x\n", p);
```

Q4

```
int increment(int *x){
    // increment recieves pointer to i
    // the value of i is incremented by 1 to 5 + 1 = 6
    // j will be 6
    // i will be 6
    return ++(*x);
}

int increment(int *x){
    // increment recieves pointer to i
    // (*x)++ will be treated as *(x++)
    // this means j will be 5 and i will be 6
    return (*x)++;
}
```

Tobias Brunner
08-102-204

Esther Hoeun Yu

Q5

```
short x[4] = {1, 2, 3, 4};
int *px = x;

//prints 1 1
printf("%i %i\n", *x, *(short *)px);
px++;
//prints 1 3
//address px-1 could already be used and will be overwritten
printf("%i %i\n", *x, *(short *)px);
```

```
short x = 3;
short *px = &x;
*(px--) = 20;
*px = 21;
// prints 20 21
printf("%i %i\n", x, *px);
```

Q6

```
struct {
    char a[10];
    char b;
    char c;
    short int d;
}myStruct;

union {
    char a[12];
    int b;
    short int d[4];
} myUnion;

// struct: each element gets its own disk space -> 14
// char a[10]: 10 bytes + char b 1 byte + char c 1 byte + short int 2 bytes
// union: disk space is defined by biggest element -> 12
// char a[12]: 12 bytes
```

Tobias Brunner
08-102-204

Esther Hoeun Yu

Q7

```c
#define callA callB(5)
/*The #define creates a macro,
 * which is the association of an identifier or parameterized identifier
 * with a token string. After the macro is defined, the compiler can
 * substitute the token string for each occurrence of the identifier
 * in the source file.
 */

void callB(int a){
    // prints 10
    printf("%i\n", a*2);
}


int main(int argc, const char * argv[]) {
    // this calls is substituted for callB(5)
    callA;
    return 0;
```