Informatics II, Spring 2023, Exercise 6

Publication of exercise: March 27, 2023
Publication of solution: April 02, 2023
Exercise classes: April 03 - April 07, 2023

Task 1

Pointer

1. What does the following code print to the console?

```
1  int main() {
2      int a = 1;
3      int *p;
4      p = &a;
5      printf("%p", p);
6
7      return 1;
8  }
```

A. 1

B. The address of p

C. The address of a

D. None of the above

2. What does the following code print to the console?

```
1  int main() {
2      int a = 1;
3
4      int *p;
5      p = malloc(sizeof(int));
6      *p = 2;
7      a = *p;
8
9      free(p);
10
11     printf("%d", a);
12
13     return 1;
14 }
```

A. 1

B. 2

3. Indicate whether the following statement is true or false:
"In the C language, function arguments are passed by value."

A. True

B. False

4. What does the following code print to the console?

```
1   void swap1(int a, int b) {
2       int temp;
3
4       temp = a;
5       a = b;
6       b = temp;
7   }
8
9   void swap2(int *a, int *b) {
10      int temp;
11
12      temp = *a;
13      *a = *b;
14      *b = temp;
15  }
16
17  int main() {
18      int a = 1;
19      int b = 2;
20      swap1(a, b);
21      printf("%d,%d,", a, b);
22
23      swap2(&a, &b);
24      printf("%d,%d", a, b);
25
26      return 1;
27  }
```

A. 1, 2, 1, 2

B. 2, 1, 2, 1

C. 2, 1, 1, 2

D. 1, 2, 2, 1

Array

5. Consider the following code and find the invalid statement

```
1   int main() {
2       int a[3] = {1, 2, 3};
3       int *p;
4       /* the following statements fit here. */
5
6       return 1;
7   }
```

A. p = a; p++;

B. p = &a[0]; p++;

C. int b[3] = a;

D. int b = *(a + 1);

6. At the end of the execution, what are the contents of a?

```
1   void swap(int *p, int id1, int id2) {
2       *(p + id1) += *(p + id2);
3       *(p + id2) = *(p + id1) - *(p + id2);
4       *(p + id1) -= *(p + id2);
5   }
6
7   int main() {
8       int a[8] = {2, 0, 2, 3, 0, 5, 2, 6};
9       swap(a, 2, 3);
10
11      return 1;
12  }
```

A. 2, 0, 2, 3, 0, 5, 2, 6

B. 2, 2, 0, 3, 0, 5, 2, 6

C. 2, 0, 3, 2, 0, 5, 2, 6

7. Whenever an array name, such as a, is passed to a function, effectively the address of its initial element is passed as a parameter, namely $\&a[0]$. Suppose the same *swap* function and array a in question 6 are given. At the end of the execution, what are the contents of a, considering the code below?

```
1   int main() {
2       swap(&a[1], 2, 3);
3
4       return 1;
5   }
```

A. 2, 0, 2, 3, 0, 5, 2, 6

B. 2, 0, 3, 2, 0, 5, 2, 6

C. 2, 0, 2, 0, 3, 5, 2, 6

Task 2

Given the following code where an integer array is initialized,

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #define N 5
4
5  void print(int *arr, int n); // (a)
6
7  int* reverse(int *arr, int n); // (b)
8
9  int* prepend(int *arr, int v); // (c)
10
11 /*Please complete implementation here*/
12
13 int main() {
14     int *arr;
15     arr = malloc(N * sizeof(int));
16
17     for (int i = 0; i < N; i++) {
18         arr[i] = i;
19     }
20
21     printf("The original: ");
22     print(arr, N);
23
24     int *reversed = reverse(arr, N);
25
26     free(arr);
27
28     printf("The reversed: ");
29     print(reversed, N);
30
31     int *prepended = prepend(reversed, 5);
32
33     free(reversed);
34
35     printf("The prepended: ");
36     print(prepended, N + 1);
37 }
```

Implement 3 functions to perform the following tasks,

a) Implement $print(int\ *arr, int\ n)$ to print all elements of an array with pointers instead of indices like $arr[n]$.

b) Implement $reverse(int\ *arr, int\ n)$ to reverse the order of the given array arr.

c) Implement $prepend(int\ *arr, int\ v)$ to prepend a given value v to the array arr.

Task 3

Consider a linked list of which a node is defined as follows together with a randomly initialized integer array of N elements,

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define N 10
4
5  struct node {
6      int key;
7      struct node* next;
8  };
9
10 struct node* convertArraytoLinkedList(int *arr, int n); // (a)
11
12 void print(struct node* curr); // (b)
13
14 struct node* reverseLinkedList(struct node* head); // (c)
15
16 int main() {
17     /* Generating an array of N random integers */
18     int *arr;
19     arr = malloc(N * sizeof(int));
20     for (int i = 0; i < N; i++) {
21         *(arr + i) = rand();
22     }
23
24     struct node *head = convertArraytoLinkedList(arr, N);
25
26     printf("\nThe original: ");
27     print(head);
28
29     struct node* newHead = reverseLinkedList(head);
30
31     printf("\nThe reversed: ");
32     print(newHead);
33
34     return 1;
35 }
```

Please implement the following functions,

a) Implement $convertArraytoLinkedList(int\ *arr, int\ n)$ to generate a linked list from a given array.

b) Implement $print(struct\ node*\ curr)$ to print a linked list.

c) Implement $reverseLinkedList(struct\ node*\ head)$ to reverse the order of a linked list and return a pointer to the new head.