J.NAWROCKI, M. ANTCZAK, W. FROHMBERG, K. KOLANOWSKI, J. POCHMARA,
S. WĄSIK, T. ŻOK

ALGORYTMY I STRUKTURY DANYCH

ZAD. 1. Rozwiąż Quiz, który dostępny jest dla tych laboratoriów na moodle'u. Upewnij się, że rozumiesz dlaczego dla każdego pytania konkretny wynik jest prawidłowy.

ZAD. 2. Zdefiniuj wartość zmiennej wynikowej dla każdej instrukcji przypisania w poniższym kodzie zakładając, że instrukcje będą uruchamiane w sposób sekwencyjny:

```
int main()
{
   char blocks[3] = {'A','B','C'};
   char *ptr = &blocks[0];
   char temp;
   temp = blocks[0];                    //.....

   temp = *(blocks + 2);                //.....

   temp = *(ptr + 1);                   //.....

   temp = *ptr;                         //.....

   ptr = blocks + 1;
   temp = *ptr;                         //.....

   temp = *(ptr + 1);                   //.....

   ptr = blocks;
   temp = *++ptr;                       //.....

   temp = ++*ptr;                       //.....

   temp = *ptr++;                       //.....

   temp = *ptr;                         //.....


   return 0;
}
```

ZAD. 3 (FENIX C1). Uzupełnij funkcję `printReverseString(char *s)` w taki sposób, aby wypisywała ona na konsolę znaki łańcucha wejściowego w odwrotnej kolejności. Podczas realizacji zadania wykorzystaj tylko i wyłącznie własności wskaźników.

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

int printReverseString(char *s) {
    char *cptr;
    …
}

int main()
{
    char s[10];
    scanf("%s", s);
    printReverseString(s);
    return 0;
}
```

Wejście: Ciąg maksimum 9 znaków, np. `asdf`
Wyjście: Odwrócony ciąg, np. `fdsa`

ZAD. 4. Uzupełnij funkcję `max(float *far, int size)` w taki sposób, aby zwracała wskaźnik na największy element tablicy liczb zmiennoprzecinkowych. Podczas realizacji zadania wykorzystaj tylko i wyłącznie własności wskaźników.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

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

float* max(float *far, int size) {
    int i;
    float max=*far;
    float *maxPtr = far;
    …
    return maxPtr;
}

int main() {
    float far[4] = {1.5f, 3.2f, 2.3f, 2.7f};
    float *fptr = max(&far,sizeof(far)/sizeof(*far));
    printf("%f",*fptr);
    return 0;
}
```

ZAD. 5. Uzupełnij funkcję `myStrLength(char *s)` w taki sposób, aby zwracała liczbę znaków łańcucha wejściowego. Podczas realizacji zadania wykorzystaj tylko i wyłącznie własności wskaźników.

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

int myStrLength(char *s) {
    int count=0;
    char *ls=s;
    …
    return count;
}

int main()
{
     char s[10] = "abcde";
     printf("%d", myStrLength(s));
}
```

ZAD. 6. Uzupełnij funkcję `contains(char *s, char c)` w taki sposób, aby zwracała 1 gdy dany znak *c* znajduje się w łańcuchu reprezentowanym przez *s* natomiast 0 w przeciwnym wypadku. Podczas realizacji zadania wykorzystaj tylko i wyłącznie własności wskaźników.

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

int contains(char *s, char c) {
    char *ls=s;
    …
    return 0;
}

int main()
{
     char s[10] = "abcde";
     printf("%d", contains(s,'e'));
}
```

ZAD. 7 (FENIX C5). Uzupełnij funkcję `revString(char *s)` w taki sposób, aby odwracała zawartość łańcucha wejściowego. Podczas realizacji zadania wykorzystaj tylko i wyłącznie własności wskaźników.

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

void revString(char *s) {
    char *s2 = s;
    char temp;
    …
}

int main()
{
     char s[100];
     scanf("%s", s);
     revString(s);
```

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

```
    printf("%s",s);
}
```

ZAD. 8. Utwórz drzewo BST, do którego wczytywane są kolejno wartości kluczy <10,3,15,17,9,12,16>.

ZAD. 9. Jak będzie wyglądało drzewo, jeżeli usunięty zostanie wierzchołek:

- 12,
- 17,
- 15.

ZAD. 10. Utwórz kopiec (kolejkę priorytetową) dla danych z zadania 8.

We wszystkich poniższych zadaniach:

1. **Opracuj zbiór przypadków testowych, które doprecyzują interfejs użytkownika.**
2. **Napisz program w C stosując statyczną strukturę tablicy (lub macierzy).**
3. **Przelicz na nim wybrane przypadki testowe metodą ręcznej symulacji.**
4. **Zmodyfikuj program w taki sposób, aby wykorzystać dynamiczną strukturę tablicy (lub macierzy).**

ZAD. 11(*). Napisz program czytania rozmiaru *n* i rysowania macierzy o rozmiarach *n* x *n* typu *prawa-górna*. Macierz *prawa-górna* zawiera na głównej przekątnej i nad tą przekątną znaki *X*, natomiast poniżej głównej przekątnej są znaki spacji. Oto przykład takiej macierzy dla *n=4*:

```
XXXX
 XXX
  XX
   X
```

ZAD. 12(*, FENIX C7). Napisz program, który będzie wczytywał od użytkownika liczby, tak długo aż nie poda on liczby -1, a następnie wypisze wczytane liczby w odwrotnej kolejności. Zakładamy, że początkowo dynamiczna tablica liczb zainicjowana jest zbiorem 10 elementów (`malloc`), która może zmieniać swój rozmiar (zmniejszać się lub zwiększać) podczas działania programu (`realloc`) gdy użytkownik będzie podawał kolejne liczby.

Wejście: Ciąg liczb o wartościach od 0 do 1’000’000, np. `9 7 11 -1`
Wyjście: Ciąg odwrócony, np. `11 7 9`

ZAD. 13(*, FENIX C8). Napisz program **scalania ciągów**, który polega na łączeniu dwóch posortowanych ciągów w jeden ciąg posortowany. Ciągi nie muszą być równych długości. Zakładamy, że użytkownik definiuje kolejne wartości liczb, które mają się znaleźć w ciągu bez określania wcześniej liczby elementów danego ciągu. Dany ciąg kończy się, gdy użytkownik poda specjalną, umowną wartość równą -1, która nie jest wliczana jako element ciągu. Każdy ciąg (tablica dynamiczna) inicjowany jest pewną stałą liczbą elementów, która następnie jest rozszerzana w razie potrzeby.

Wejście: Dwa ciągi liczb o wartościach od 0 do 1’000’000, np. `3 5 10 -1 2 6 -1`
Wyjście: Scalony ciąg, np. `2 3 5 6 10`

ZAD. 14(*). Napisz program **mnożenia macierzy kwadratowych**.

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.

ZAD. 15(*, FENIX C10). Napisz program sortujący ciąg liczb naturalnych metodą przez **wymianę/wybór** (selectionsort).

Wejście: Liczba elementów do posortowania od 1 do 1000, a następnie elementy o wartościach od 0 do 1’000’000, np. `5 3 7 2 15 14`
Wyjście: Posortowany ciąg, np. `2 3 7 14 15`

(*) gwiazdką oznaczone są zadania, które nie są realizowane na ćwiczeniach i są przeznaczone do wykonania jako zadania domowe.