Algorytmy 3.

Konstrukcja wybranych algorytmów - cd.

1. Algorytm sprawdzający porządek niemalejący w tablicy a[N]
   a. wersja z break (niestrukturalna)

```
int a[N]; int i = 0;
        while( i < N-1 ) {
              if( a[i] > a[i+1] ) break;
              else i++;
         }
         if( i == N-1) cout<< "Tablica jest uporządkowana"<<endl;
         else    cout<< "Tablica nie jest uporządkowana"<<endl;
```

   b. wersja strukturalna

```
 int a[N]; int i = 0;
        while( i <N-1 && a[i]<=a[i+1])   i++;
         if( i==N-1) cout<< "Tablica jest uporządkowana"<<endl;
         else    cout<< "Tablica nie jest uporządkowana"<<endl;
```

   c. wersja z wartownikiem:

```
int a[N+1];
a[N] = a[N-1] - 1;   // wartownik a[N-1]> a[N]
i=0;
while(a[i] <= a[i+1])       i++;
if( i==N-1) cout<< "Tablica jest uporządkowana"<<endl;
else    cout<< "Tablica nie jest uporządkowana"<<endl;
```

2. Algorytmy usuwające z tablicy zadany element:
   a. x – wartość elementu, który ma być usunięty, przy czym usuwany jest tylko pierwsze wystąpienie x, jeśli x nie występuje w tablicy - algorytm nic nie robi.

```
int a[100]; int i;
int  N;    // N- aktualna długość tablicy
for(i= 0; i < N; i++) {
  if( a[i] == x ) break;
}
if( i != N ) { // x == a[i] więc go usuwamy
   for(int j = i; j < N-1; j++){
     a[j] = a[j+1]; // przesuwamy elementy w lewo
   }
   N--; // po usunięciu zmniejszamy liczbę elementów w a
}
```

   b. usuwa element z pozycji ‘p’

```
if(p<0 && p >=n-1)
    cout << "Zła pozycja"<< endl;
else {
        for(int j = p;  j < N-1;  j++){
            a[j] = a[j+1]; // przesuwamy elementy w lewo
        }
        N--;  // po usunięciu zmniejszamy liczbę elementów w a
}
```

3. Algorytm, który wstawia do tablicy uporządkowanej niemalejąco element x zachowując porządek w tablicy.

```
int a[100]; int i;
int x,  // wstawiany element
    N; // N- aktualna długość tablicy

    if( N == 100 ) cout << "Tablica pełna"<< endl;
    else
    {     i = N-1;
          while( i >= 0   &&  x < a[i])
          {
             a[ i+1 ] = a[ i ];  // przesunięcie w prawo
             i = i-1 ;
          }
          a[ i+1 ] = x;          // można wstawić  x
          N++;                  // po wstawieniu zwiększamy liczbę elementów  tablicy
         }
    }
```

4. Algorytm obliczający wartość wielomianu *W(x)* stopnia n dla zadanego x;

$$W(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

Skorzystamy z postaci iloczynowej wielomianu //schemat Hornera:

$W(x) = (\ldots.(a_n x + a_{n-1})\, x + \ldots + a_1)\, x + a_0$

```
float a[n+1]; // współczynniki wielomianu
float x, w;
 ....
w=a[n];
for(int i = n; i > 0; i--)
  w=w*x+a[i-1]
```

Zadania.

1. Napisz algorytm, który wstawia zadany element do tablicy na zadaną pozycję.
2. Napisz algorytm wyszukiwania binarnego zadanego elementu w tablicy uporządkowanej.
3. Napisz algorytm znajdujący najdłuższą podtablicę niemalejącą w tablicy a[N].
4. Jakie wartości x, z wypisze poniższy algorytm:

```
int  x =90, y=230, z=0;
while( x < y ) {
 z++;
 if(z % 2 == 1) x = x+3;
 else x = x+4;
 y = y-2;
}
pisz(x, z);
```

5. Dany jest ciąg licz całkowitych a[0],...,a[n-1], przy czym: n> 4, który nie zawiera duplikatów oraz w ciągu są co najmniej dwa elementy parzyste i dwa elementy nieparzyste.
   Napisz algorytm oblicza dwie liczby całkowite:
   - x- wartość najmniejszego elementu o numerze nieparzystym
   - y - numer największego elementu o wartości parzystej.

6. Dopisz takie instrukcje w miejsca ... aby algorytm podstawił pod zmienna Y wartość 1, jeśli X jest liczba pierwsza lub 0 w przeciwnym przypadku

```
WCZYTAJ(X) ;
Y  =  ..........  ;
I  =  2 ;
do {
.....................................................
.....................................................
} while(.....................) && (.....................)
```

7. **Dopisz takie instrukcje w miejsca ... aby algorytm wprowadził do tablicy nie więcej niż 100 liczb losowych i zakończył losowanie, gdy kolejna liczba będzie zero.**

```
int N, L;   // N - liczba losowanych elementów
            // L – pomocnicza zmienna
N = ..........;
LOSUJ ( L ) ;
while (.....................) {
    ..............................  ;
    ..............................  ;
   LOSUJ( L );
}
WYPISZ ("Liczba wylosowanych liczb= ", N)
```