1. (6 поена) а) Објаснити декларацију std::stack<int> mystack. Наведите кључне операције које се могу вршити над објектом mystack и њихову временску и просторну сложеност.
б) Написати програм који са стандардног улаза уноси ниску карактера и исписује 1 уколико су у њој правилно упарене три стандардне врсте заграда, а 0 ако нису. Алгоритам треба да буде линеарне сложености у односу на дужину ниске.

Улаз	Излаз
aa(bbb[cc)ddd]e	0

Решење:

b)

```
#include <iostream>

#include <stack>

using namespace std;

stack<char> s;

int main()

{

   char c;

   while(cin.get(c))

   {

      /* Ako je ucitana otvorena zagrada, stavlja se na stek */

   if (c == '(' || c == '{' || c == '[')

         s.push(c);

   /* Ako je ucitana zatvorena zagrada, proverava se da li je stek

      prazan i ako nije, da li se na vrhu steka nalazi odgovarajuca

      otvorena zagrada */

   else {

     if (c == ')' || c == '}' || c == ']') {

       if (!s.empty()  && ((s.top() == '(' && c == ')')

                        || (s.top() == '{' && c == '}')

                        || (s.top() == '[' && c == ']'))) {
```

```
        /* Sa vrha steka se uklanja otvorena zagrada */
        s.pop();
       } else {
        /* Dakle, zagrade u izrazu nisu ispravno uparene */
        break;
       }
     }
   }
 }

/* Ako je stek prazan i procitana je cela datoteka, zagrade su
    ispravno uparene. */
 if (s.empty()  && cin.fail())
   cout << "1\n";  // Zagrade su ispravno uparene
 else
   /* U suprotnom se zakljucuje da zagrade nisu ispravno uparene. */
   cout <<" 0\n";  // Zagrade nisu ispravno uparene.
   return 0;
}
```

2. (7 поена) Написати програм који са стандардног улаза уноси позитиван цео број N и N реалних строго позитивних бројева x_1, x_2..., x_n и на стандардни излаз исписује производ сегмената x_i, x_{i+1},..., x_j (1 <= i<= j <= n) са највећим могућим производом. Производ празног сегмента је по дефиницији једнак 1. Алгоритам треба да буде линеарне сложености у односу на дужину низа.

Улаз	Излаз
8	120
2.1 4 0.1 4 5 6 0.01 9.2	

Решење: Директна примена Кадановог алгоритма

Да би се добио максималан број поена, потребно је и образложити временску слоежност добијеног решења.

3. (8 поена) Написати програм који са стандардног улаза уноси позитивне целе бројеве N и M, потом уноси низ од N целих бројева, а потом M парова бројева i, j таквих да важи 0 <= i <= j < N и за сваки од парова исписује суму свих елемената низа са индексима у интервалу [i, j]. Алгоритам треба да буде сложености O(N+M).

Улаз	Излаз
10 3	3
1 2 3 4 5 6 7 8 9 10	30
2 2	15
3 7	
0 4	

Решење:

Директан и мање ефикасан начин је да се након учитавања низа за сваки сегмент збир изнова рачуна у петљи. Ако низ има n елемената и желимо да израчунамо збирове његових m сегмената, укупна сложеност била би $O(n \cdot m)$.

Један од начина да се задатак ефикасно реши је то да се примети да се сваки збир сегмента може представити као разлика два збира префикса низа. Наиме, збир сегмента одређеног позицијама из интервала $[l, d]$ једнак је:

$$\sum_{i=l}^{d} a_i = \sum_{i=0}^{d} a_i - \sum_{i=0}^{l-1} a_i$$

Дакле, ако знамо збирове свих префикса, збир сваког сегмента можемо израчунати у времену $O(1)$. Збирове префикса, наравно, можемо рачунати инкрементално, тако да је време потребно за њихово израчунавање $O(n)$. Укупна сложеност је онда $O(n + m)$.

```
#include <iostream>
using namespace std;
int main()
{
   int n, m;
   cin >> n>>m;

// zbirPrefiksa[i] - zbir elemenata iz intervala [0, i)
  vector<int> zbirPrefiksa(n+1);
  zbirPrefiksa[0] = 0;

for (int i = 0; i < n; i++) {
      int x;
      cin >> x;
      zbirPrefiksa[i+1] = zbirPrefiksa[i] + x;
}

for (int i = 0; i < m; i++) {
```

```
  int l, d;
   cin >> l >> d;
   cout << zbirPrefiksa[d+1] - zbirPrefiksa[l] << endl;
}

  return 0;
}
```

Претпроцесирање је у овом примеру подразумевало израчунавање помоћних података који су омогућили ефикасније касније извршавање алгоритма