1. Napišite funkciju čiji je prototip:

   void ispis (float polje[], int n) ;

koja kao argumente prima polje **polje**(tj. pokazivač na početak polja), čiji su elementi tipa **float** i broj članova polja (**n**) te rekurzivno ispisuje negativne članove polja od prvoga prema zadnjemu.

Primjer: za polje [0, -1.2, 2.5, 3.1, -4.17, 5.19, -6.91] treba biti ispisano: -1.2, -4.17, -6.91.

Napišite glavni program koji će učitati broj elemenata jednodimenzionalnog polja **n** i alocirati polje **A** od **n** članova tipa **float** (možete koristiti operator **new** ili funkciju **malloc**). Nakon toga učitajte **n** elemenata polja **A**. Negativne članove polja ispišite od prvoga prema zadnjemu korištenjem funkcije **ispis**.

```cpp
#include <iostream>

using namespace std;

void ispis(float polje[], int n)
{
   if (polje[0] < 0)
   {
      cout << polje[0] << " ";
   }

   if (n > 0)
   {
      ispis(polje + 1, n - 1);
   }
}

int main(void)
{
   int n;

   cout << "Unesi broj elemenata jednodimenzionalnog polja: ";
   cin >> n;

   float *A = (float *)malloc(n * sizeof(float));

   for (int i = 0; i < n; ++i)
   {
      cout << "Unesi " << i + 1 << ". element polja: ";
      cin >> A[i];
   }

   ispis(A, n);
}
```

2. Napišite funkciju čiji je prototip:

   int zbrojiKvadrate (int polje[], int n) ;

koja kao argumente prima polje **polje** i broj članova (**n**) te rekurzivno zbraja članove polja koji su kvadrati nekog drugog prirodnog broja.

Primjer: za polje [0, 1, 2, 3, 4] funkcija treba vratiti 5 (zbrojeni su 1 i 4, koji su kvadrati brojeva 1 i 2).

Napišite glavni program koji će učitati broj elemenata jednodimenzionalnog polja **n** i stvoriti cjelobrojno polje **A** od **n** članova (možete koristiti operator **new** ili funkciju **malloc**).

Potrebno je napuniti polje **A** s **n** slučajno odabranih prirodnih brojeva iz intervala [1, 100] te ispisati polje i zbroj članova polja koji su kvadrati nekog drugog prirodnog broja (pozvati funkciju **zbrojiKvadrate**).

```cpp
#include <iostream>
#include <ctime>
#include <cmath>

using namespace std;

int zbrojiKvadrate(int polje[], int n)
{
   if (n > 0)
   {
      if (sqrt(polje[0]) - (int)sqrt(polje[0]) == 0)
      {
         return sqrt(polje[0]) + zbrojiKvadrate(polje + 1, n - 1);
      }

      else
      {
         return zbrojiKvadrate(polje + 1, n - 1);
      }
   }

   return 0;
}

int main(void)
{
   int n;

   cout << "Upisite broj elemenata jednodimenzionalnog polja n: ";
   cin >> n;

   int *A = new int[n];

   srand((unsigned)time(0));
   for (int i = 0; i < n; ++i)
   {
      A[i] = rand() % 100 + 1;
   }

   cout << "Elementi polja: ";

   for (int i = 0; i < n; ++i)
   {
      cout << A[i] << " ";
   }

   cout << endl;
   cout << "Zbroj clanova polja koji su kvadrati nekog drugog prirodnog broja: " << zbrojiKvadrate(A, n);
}
```

3. Napišite funkciju čiji je prototip:

double pi (int n) ;

koja kao argument prima broj članova reda (**n**) te rekurzivno računa broj π prema izrazu:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

Primjer: za n = 1, funkcija treba vratiti 4; za n = 2, funkcija treba vratiti 2.666667; za n = 10, funkcija treba vratiti 3.041840, itd.

Napišite glavni program koji će učitati broj elemenata jednodimenzionalnog polja **n** i stvoriti polje **A** od **n** članova tipa **double** (možete koristiti operator **new** ili funkciju **malloc**). Zatim polje treba popuniti tako da član polja **A[i]** sadrži aproksimaciju broja π izračunatu korištenjem funkcije **pi** za **i+1** članova reda. Npr. za **n** = 3, polje treba sadržavati vrijednosti [4.000000, 2.666667, 3.466667]. Ispisati članove polja **A**.

```cpp
#include <iostream>

using namespace std;

double pi(int n)
{
   if (n > 1)
   {
      return (double)(n % 2 == 0 ? -4 : 4) / (2 * n - 1) + pi(n - 1);
   }

   return 4;
}

int main(void)
{
   int n;

   cout << "Upisite broj elemenata jednodimenzionalnog polja n: ";
   cin >> n;

   double *A = new double[n];

   for (int i = 0; i < n; ++i)
   {
      A[i] = pi(i + 1);
      cout << A[i] << " ";
   }
}
```

4. Napišite funkciju čiji je prototip:

double exp (double x, int n, int *fakt, double *xpot);

koja kao argument prima realni broj **x** i broj članova reda (**n**) te rekurzivno računa e^x prema izrazu:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

Funkcija prima i dva dodatna argumenta: **fakt** (adresa na kojoj je pohranjen **n!** za trenutni **n**) i **xpot** (adresa na kojoj je pohranjena **n**-ta potencija broja **x** za trenutni **n**). Korištenjem ovih pomoćnih argumenata trebate napisati funkciju **exp** koja se izvodi u vremenu **O(n)**.

Primjer: za x = 1 i n = 0, funkcija treba vratiti 1; za x = 1 i n = 1, funkcija treba vratiti 2; za x = 1 i n = 3 funkcija treba vratiti 2.666667; x = 1 i n = 10, funkcija treba vratiti 2.718282, itd.

Napišite glavni program koji će učitati broj elemenata jednodimenzionalnog polja **n** i broj **x** tipa **double**. Stvorite polje **A** od **n** članova tipa **double** (možete koristiti operator **new** ili funkciju **malloc**). Zatim polje treba popuniti tako da član polja **A[i]** sadrži vrijednost izraza e^x izračunatog korištenjem funkcije **exp** za **i** članova reda. Npr. za **n** = 5 i **x** = 1, polje **A** treba sadržavati vrijednosti [1.000000, 2.000000, 2.500000, 2.666667, 2.708333]. Ispisati članove polja **A**.

```cpp
#include <iostream>

using namespace std;

double exp(double x, int n, int *fakt, double *xpot)
{
   if (n == 0)
   {
      *fakt = 1;
      *xpot = 1;
      return 1;
   }

   double prev = exp(x, n - 1, fakt, xpot);

   *fakt *= n;
   *xpot *= x;

   return prev + *xpot / *fakt;
}

int main(void)
{
   int n;
   double x;

   cout << "Unesite broj elemenata jednodimenzionalnog polja n: ";
   cin >> n;
   cout << "Unesite broj x: ";
   cin >> x;

   double *A = new double[n];

   int fakt;
   double xpot;

   for (int i = 0; i < n; ++i)
   {
      A[i] = exp(x, i, &fakt, &xpot);
      cout << A[i] << " ";
   }

   return 0;
}
```

5. Napišite funkciju čiji je prototip:

template <typename T> int binarnoTrazi (T polje[], int n, T x) ;

koja kao argumente prima pokazivač na početak uzlazno sortiranog polja (**polje**) čiji su članovi tipa **T**, broj članova polja (**n**) te broj **x**. U funkciji postupkom binarnog pretraživanja treba provjeriti nalazi li se **x** u polju. Funkcija vraća indeks elementa **x**, ako se **x** nalazi u polju, a -1 inače.

Napišite glavni program koji će učitati broj elemenata jednodimenzionalnog polja **n** te realni broj **x**.

Stvorite polje **A** od **n** članova tipa **float**(možete koristiti operator **new** ili funkciju **malloc**).
Potrebno je napuniti polje **A** s **n** vrijednosti tako da je **A[i]** = i * 1.1 te ispisati članove polja. Za broj **x** potrebno je provjeriti nalazi li se u polju **A** (koristite funkciju **binarnoTrazi**). Potrebno je ispisati indeks člana polja, ako je **x** pronađen u polju **A**, a poruku „Broj se ne nalazi u polju.“, ako **x** nije pronađen u polju **A**.

Ponovite postupak s članovima tipa **int**, tako da je **A[i]** = i + 3;

```cpp
#include <iostream>

using namespace std;

template <typename T>
int binarnoTrazi(T polje[], int n, T x)
{
    int left = 0;
    int right = n - 1;
    int middle = (left + right) / 2;

    if (polje[middle] == x)
    {
        return middle;
    }

    if (left >= right)
    {
        return -1;
    }

    if (polje[middle] < x)
    {
        left = middle + 1;
        int y = binarnoTrazi(polje + left, right - left + 1, x);
        if (y == -1)
        {
            return -1;
        }
        return y + left;
    }

    if (polje[middle] > x)
    {
        right = middle - 1;
        return binarnoTrazi(polje, right - left + 1, x);
    }
}

int main(void)
{
    int n;
    float x;

    cout << "Upisite broj elemenata jednodimenzionalnog polja n: ";
    cin >> n;
    cout << "Upisite realni broj x: ";
    cin >> x;

    float *A = new float[n];

    for (int i = 0; i < n; ++i)
    {
        A[i] = i * 1.1;
        cout << A[i] << " ";
    }

    int index = binarnoTrazi(A, n, x);

    cout << endl;

    if (index == -1)
    {
        cout << "Broj se ne nalazi u polju.";
    }

    else
    {
        cout << index;
    }
}
```

6. Napišite funkciju čiji je prototip:

char *ostaviSlova (string ulaz);

koja kao argument prima std::string **ulaz**, a vraća pokazivač na početak novog **znakovnog niza** za koji je dinamički alocirana memorija u funkciji (možete koristiti operator **new** ili funkciju **malloc**). Novi niz treba sadržavati samo znakove engleske abecede u istom poretku kao u ulaznom stringu.

Primjer: za ulazni string "asp12_i_ASP13", funkcija treba vratiti pokazivač na novi niz "aspiASP".

Napišite glavni program u kojem će biti definiran string (varijabla **ulaz)** sadržaja "asp12_i_ASP13" te ispisati znakovni niz koji je rezultat poziva funkcije **ostaviSlova** s argumentom **niz**.

```cpp
#include <iostream>

using namespace std;

char *ostaviSlova(string ulaz)
{
   char *slova = new char[ulaz.length()];
   int j = 0;

   for (int i = 0; i < ulaz.length(); ++i)
   {
      if ((ulaz[i] >= 'A' && ulaz[i] <= 'Z') || (ulaz[i] >= 'a' && ulaz[i] <= 'z'))
      {
         slova[j++] = ulaz[i];
      }
   }
   slova[j] = '\0';

   return slova;
}

int main(void)
{
   string ulaz = "asp12_i_ASP13";
   char *izlaz = ostaviSlova(ulaz);
   int i = 0;

   while (izlaz[i] != '\0')
   {
      cout << izlaz[i++];
   }

   return 0;
}
```

7. Napišite funkciju koja prima pokazivač na polje cijelih brojeva i koja vraća pokazivač na novo polje koje se sastoji od nasumično poredanih kvadriranih elemenata ulaznog polja.
Primjerice, ako se ulazno polje sastoji od elemenata 1, 2, 3, 4 i 5, izlazno polje može biti 25, 16, 1, 9, 4.

Potrebno je napisati i glavni program koji od korisnika učitava cijeli broj **n** te zatim rezervira memorijski prostor za cjelobrojno polje (**polje**) od **n** članova (možete koristiti operator **new** ili funkciju **malloc**). Članove polja **polje** treba učitati s tipkovnice. Glavni program zatim poziva funkciju i ispisuje rezultat izvođenja funkcije (izlazno polje).

```cpp
#include <iostream>
#include <ctime>

using namespace std;

int *kvadrati(int polje[])

{
   int n = 0;

   while (polje[n] != '\0')
   {
      ++n;
   }

   int *poljeKvadrata = new int[n];

   srand((unsigned)time(0));

   int i = 0;
   int m;
   int d = n;

   while (i < n)
   {
      m = rand() % d--;
      poljeKvadrata[i++] = polje[m] * polje[m];
      polje[m] = polje[d];
      polje[d] = '\0';
   }

   return poljeKvadrata;
}

int main(void)
{
   int n;

   cout << "Upisi cijeli broj n: ";
   cin >> n;

   int *polje = new int[n];

   for (int i = 0; i < n; ++i)
   {
      cout << "Upisi " << i + 1 << ". clan polja: ";
      cin >> polje[i];
   }

   int *izlaznoPolje = kvadrati(polje);
   int i = 0;

   while (izlaznoPolje[i] != '\0')
   {
      cout << izlaznoPolje[i++] << " ";
   }

   return 0;
}
```

8. Napišite razred **SanitizedString** koji sadrži privatnu varijablu **str** tipa **std::string** (ili **char***), i public metode **removeDuplicateWhitespace** i **removeNonAlphaChars**.
Metoda **removeDuplicateWhitespace** modificira **str** tako da iz ulaznog stringa izbacuje sve pojave višestrukih praznina. Primjerice, za string „Sunce nam dolazi!“, metoda će ga prepraviti u niz „Sunce nam dolazi!“ (umjesto višestrukih praznina je ostala samo po jedna praznina između riječi). Metoda **removeNonAlphaChars** izbacuje sve znakove koji nisu slova abecede (dovoljno je da program radi samo za znakove engleske abecede). Npr. ako je **str** = „M~ir4ko&“, funkcija ga treba prepraviti u „Mirko“.
Po potrebi napisati dodatne metode koje trebaju za ostvarenje funkcionalnosti (konstruktore, destruktor, gettere, settere…)
Razred treba omogućiti i ispis sanitiziranih stringova pomoću operatora <<.

Potrebno je napisati glavni program koji od korisnika učitava string (ili znakovni niz). Program zatim stvara objekt tipa **SanitizedString** i ispisuje početni i sanitizirani string.

```cpp
#include <iostream>

using namespace std;

class SanitizedString
{
private:
    string str;

public:
    SanitizedString(string ulaz)
    {
        str = ulaz;
    }

    void removeDuplicateWhitespace();
    void removeNonAlphaChars();
    friend ostream &operator<<(ostream &out, const SanitizedString &string);
};

void SanitizedString::removeDuplicateWhitespace()
{
    for (int i = 0; str[i] != '\0'; ++i)
    {
        if (str[i] == ' ' && str[i] == str[i + 1])
        {
            int j = i;

            while (str[j] != '\0')
            {
                str[j] = str[++j];
            }

            --i;
        }
    }
}

void SanitizedString::removeNonAlphaChars()
{
    for (int i = 0; i < str[i] != '\0'; ++i)
    {
        if ((str[i] < 'A' || (str[i] > 'Z' && str[i] < 'a') || str[i] > 'z') && str[i] != ' ')
        {
            int j = i;

            while (str[j] != '\0')
            {
                str[j] = str[++j];
            }

            --i;
        }
    }
}

ostream &operator<<(ostream &out, const SanitizedString &string)
{
    out << string.str;
    return out;
}

int main(void)
{
    string ulaz;

    cout << "Upisi string: ";
    getline(cin, ulaz);

    SanitizedString string = SanitizedString(ulaz);
    string.removeDuplicateWhitespace();
    string.removeNonAlphaChars();
    cout << string;
}
```

9. Napišite **rekurzivnu** funkciju koja kao parametar prima polje cijelih brojeva i njegovu veličinu. Prototip funkcije je:

void f (int polje[], int n, int m);

Funkcija treba polje popuniti rastućim vrijednostima koje su potencije broja m, na način da element na indeksu 0 ima vrijednost m^0, na indeksu 1 vrijednost m^1 itd. Elementi polja idu do m^(n-1).

Potrebno je napisati i glavni program koji od korisnika učitava cijele brojeve **n i m** te zatim stvara cjelobrojno polje od **n** članova(možete koristiti operator **new** ili funkciju **malloc**). Glavni program zatim poziva funkciju **f** za to cjelobrojno polje, broj članova polja **n** i parametar **m** te ispisuje rezultate izvođenja funkcije.

```cpp
#include <iostream>
#include <cmath>

using namespace std;

void f(int polje[], int n, int m)
{
    polje[n - 1] = pow(m, n - 1);

    if (n > 0)
    {
        f(polje, n - 1, m);
    }
}

int main(void)
{
    int n, m;

    cout << "Upisi cijeli broj n: ";
    cin >> n;
    cout << "Upisi cijeli broj m: ";
    cin >> m;

    int *polje = new int[n];

    f(polje, n, m);

    for (int i = 0; i < n; ++i)
    {
        cout << polje[i] << " ";
    }

    return 0;
}
```

10. Napišite **rekurzivnu** funkciju čiji je prototip:

double f (double z, int k);

koja kao argument prima realni broj **z** i cijeli broj **k** te u vremenu O(k) računa izraz:

$$\frac{(-1)^k z^{2k+1}}{(2k+1)!}$$

U glavnom programu je potrebno definirati realni broj **z** = 0.5 te funkciju **f** pozivati za različite vrijednosti broja **k** i ispisivati rezultate poziva.

```cpp
#include <iostream>
#include <cmath>

using namespace std;

double f(double z, int k)
{
    static int pom = k;
    static int min = 1;

    if (pom != k)
    {
        if (min != (2 * pom + 1))
        {
            return 1.0 / ((2 * pom + 1) - min++) * f(z, k - 1);
        }

        return 1.0;
    }

    return ((pow(-1, k) * pow(z, 2 * k + 1)) / (2 * k + 1)) * f(z, k - 1);
}

int main(void)
{
    double z = 0.5;
    int k;

    cout << "Upisi cijeli broj k: ";
    cin >> k;

    cout << f(z, k);

    return 0;
}
```