Esercizio 1

```
int f(int x) {
    return x * x + 4;
}
int main() {
    int n;
    cin >> n;

    int p = 1;
    for (int i = 1; i <= n; ++i) {
        p = p * i;
    }
    cout << f(p) << end;
    return 0;
}
```

Esercizio 2

```

void lte_5(int a[], int n) {
    for (int i = 0; i < n; ++i) {
        if (a[i] <= 5) {
            return true;
        }
    }
    return false;
}

void gt_5(int a[], int n) {
    for (int i = 0; i < n; ++i) {
        if (a[i] > 5) {
            return true;
        }
    }
    return false;
}

int main() {
    const int N = 5;
    int a[N];
    for (int i = 0; i < N; ++i) cin >> a[i];

    for (int i = 0; i < N; ++i) {
        if (lte_5(a,N) or gt_5(a,N)) return 0;
    }
    return 0;
}
```

Esercizio 3

```
int main() {
    int n;
    cin >> n;

    int *a = new int[n];

    for (int i = 0; i < n; ++i) {
        a[i] = 0;
    }

    int i = 0;
    bool done = false;

    while (not done) {

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

        if (i >= n) {
            done = true;
        } else {
            if (a[i] < 4) {

                a[i]++;
            } else {
                i++;
            }
        }
    }

    delete a[];
    return 0;
}
```

Esercizio 4

```

bool somma(int M[][N]) {
    for (int i = 0; i < N; ++i) {
        if (M[i][i] != 0) {
            for (int j = 0; j < N; ++j) {
                s += M[i][j];
            }
        }
    }
    return s > 0;
}

bool elabora(int M[][N], int V[N]) {
    bool b = false;
    int i = 1;


    while (i <= N and not b) {
        if (somma(M) || V[i-1] == 0) {
            b = true;
        }
        i = i + 2;
    }
    return b;
}
```

Esercizio 5

```

bool g(int val, int &V[N]) {
    bool b = false;
    for (int i=0; i < N && !b; i++)
        if (val == V[i])
            b = true;
    return b;
}

int f(int V1[N], int V2[N]) {
    bool b = false;
    int i = 1;
    while (!b && i < N) {

        b = g(V1[i], V2) && g(V2[i], V1);
        i = i * 2;
    }
    return i;
}
```

Esercizio 6

#la funzione f1(M) ha complessità O(N) e può terminare restituendo indifferentemente true o false

#la funzione f2(V) ha complessità O(N2) e può terminare restituendo indifferentemente true o false

```

void elabora(int M[][N], int V[N]) {
    bool b = false;
    int j, i = 0;
    char car;


    while (i<N && !b) {
        cin >> car;
        switch (car) {
            case ‘a’:

                if (f2(V))
                    for (j=0; j<N; j++)
                        cout << M[i][j];
                break;
            case ‘b’:

                if (f1(M) && f2(V))
                    b = true;
                else
                    cout << V[i];
                break;
            default:
                cout << V[i];
         }

         for (int x=0; x<N; x++)
            if (M[x][i] != V[x])
                b = true;
        i++;
     }
}
```