# 4. Program set 3

## 1. References example

```cpp
#include <iostream>
#include <string>

// by value: does NOT affect variable given as input at called location
void bump_by_value(int x)
{
    x += 10;
}

// by reference: DOES affect variable given as input at called location
void bump_by_ref(int& x)
{
    x += 10;
}

// returns a temporary string (rvalue)
std::string make_message()
{
    return std::string("hello, temporary");
}

int main()
{
    // 1) A reference is an ALIAS for an existing object
    std::cout << "\n==== " << "Alias behavior" << " ====\n";
    int a = 5;
    int& r = a;      // r MUST be initialized; r refers to 'a'
    std::cout << "a=" << a << ", r=" << r << "\n";
    r = 7;           // writing via r modifies 'a'
    std::cout << "after r=7 -> a=" << a << ", r=" << r << "\n";
    std::cout << "&a=" << &a << ", &r=" << &r << " (same address)\n";

    // 2) References are NOT reseatable
    std::cout << "\n==== " << "Not reseatable (assignment changes the referent)" << " ====\n";
    int b = 42;
    r = b;          // This does NOT rebind r to b;
                    //  r is still bound to a;
                    // instead, r = b assigns b's value into a
    std::cout << "after r=b -> a=" << a << " (now 42), b=" << b
              << ", &r still=" << &r << ", &a=" << &a << " &b=" << &b
              << "\n";

    // 3) Pass-by-value vs pass-by-reference
    std::cout << "\n==== " << "Pass-by-value vs pass-by-reference" << " ====\n";
    int v = 10;
    bump_by_value(v);
    std::cout << "after bump_by_value(v): v=" << v << " (unchanged)\n";
    bump_by_ref(v);
    std::cout << "after bump_by_ref(v):   v=" << v << " (modified)\n";

    // 4) Const reference can bind to temporaries; lifetime is extended
```

```
    std::cout << "\n==== " << "Const reference to a temporary (lifetime extension)" << " ====\n";
    const std::string& msg = make_message(); // binds to temporary string
    std::cout << "msg = " << msg << "\n";    // valid:lifetime of temporary variable is extended to
    end of scope
    // msg[0] = 'H'; // ERROR if uncommented: const reference is read-only

    // 5) Const reference can also bind to lvalues (still an alias)
    std::cout << "\n==== " << "Const reference to an lvalue" << " ====\n";
    const int& cr = a;     // cr refers to 'a' (read-only view)
    std::cout << "a=" << a << ", cr=" << cr << "\n";
    a = 99;                // change the original...
    std::cout << "after a=99 -> cr sees " << cr << " (still refers to a)\n";

    // 6) Reference to an array element
    std::cout << "\n==== " << "Reference to an array element" << " ====\n";
    int arr[3] = {1, 2, 3};
    int& mid = arr[1];     // alias the middle element
    mid = 20;              // modifies arr[1]
    std::cout << "arr: " << arr[0] << " " << arr[1] << " " << arr[2] << "\n";

    // 7) Const reference can bind to a literal (temporary int)
    std::cout << "\n==== " << "Const reference binding to a literal" << " ====\n";
    const int& k = 123;    // binds to a temporary; lifetime extended
    std::cout << "k=" << k << "\n";

    return 0;
}
```

---

## 2. Banner printer

```
#include <iostream>

void banner(const std::string& title); // prototype

int main()
{
    banner("Welcome to the Function Show!");
    banner("Let's begin");

    return 0;
}

void banner(const std::string& title)
{
    std::cout << "==== " << title << " ====\n";
}
```

---

## 3. Factorial from a function

```cpp
#include <iostream>

int factorial(int n)
{
    int f = 1;
    for (int i = 2; i <= n; ++i)
        f *= i;

    return f;
}

int main()
{
    int n;

    std::cout << "Enter n: ";
    std::cin >> n;
    std::cout << n << "! = " << factorial(n) << std::endl;

    return 0;
}
```

---

## 4. Grade calculator from function

```cpp
#include <iostream>

char letter(int s)
{
    if (s >= 90)
        return 'A';
    if (s >= 80)
        return 'B';
    if (s >= 70)
        return 'C';
    if (s >= 60)
        return 'D';

    return 'F';
}

int main()
{
    int s;
    std::cout << "score: ";
    std::cin >> s;

    std::cout << letter(s) << std::endl;

    return 0;
}
```

## 5. Prime number checker

```cpp
#include <iostream>

bool isPrime(int x)
{
    if (x < 2)
        return false;
    if (x % 2 == 0)
        return x == 2;
    for (int d = 3; d*d <= x; d += 2)
        if (x %d ==0)
            return false;

    return true;
}

int main()
{
    int n;

    std::cout << "N: ";
    std::cin >> n;

    std::cout<<( isPrime(n) ? "prime\n" : "not prime\n" );

    return 0;
}
```

---

## 6. Repeat a string n times

```cpp
#include <iostream>
#include <string>

std::string repeat(const std::string& s, int n)
{
    std::string out;
    for (int i=0; i<n; ++i)
        out += s;

    return out;
}

int main()
{
    std::string s;
    int n;

    std::cout << "Enter word and n: ";
    std::cin >>s >> n;
```

```
    std::cout << repeat(s,n) << std::endl;

    return 0;
}
```

---

## 7. sum values in a long int data type

```
#include <iostream>

long sumToN(int n)
{
    long acc = 0;

    for (int i=1; i<=n; ++i)
        acc += i;

    return acc;
}

int main()
{
    int n;

    std::cout << "Enter n: ";
    std::cin >> n;

    std::cout << sumToN(n) << std::endl;

    return 0;
}
```

---

## 8. Find sum of digits

```
#include <iostream>

unsigned sumDigits(unsigned long x)
{
    if (x==0)
        return 1;

    unsigned sum=0;
    while (x)
    {
        sum += x % 10;
        x /= 10;
    }

    return sum;
```

```
}

int main()
{
    unsigned long x;

    std::cout << "Enter x: ";
    std::cin >> x;

    std::cout << sumDigits(x) << std::endl;

    return 0;
}
```

---

## 9. Array basics - initialize arrays 1

```
#include <iostream>

int main()
{
    int a[] = {10, 20, 30, 40};
             // size deduced = 4
    for (int i = 0; i < 4; ++i)
        std::cout << a[i] << ( i==3 ? '\n' :' ');

    return 0;
}
```

---

## 10. Array basics - initialize arrays 2

```
#include <iostream>

int main()
{
    int a[5] = {1, 2};  // -> {1,2,0,0,0}

    for (int i = 0; i < 5; ++i)
        std::cout << a[i] << ( i==4 ? '\n' : ' ' );

    return 0;
}
```

---

## 11. Array basics - initialize arrays 3

```
#include <iostream>

int main()
```

```
{
    int a[8] = {0}; // every element = 0

    for (int i = 0; i < 8; ++i)
        std::cout << a[i] << ( i==7 ? '\n' : ' ' );

    return 0;
}
```

---

## 12. Array basics - initialize arrays 4

```
#include <iostream>

int main()
{
    const int N = 5;
    int a[N];

    std::cout << "Enter 5 integers: ";
    for (int i = 0; i < N; ++i)
        std::cin >> a[i];

    for (int i = 0; i < N; ++i)
        std::cout << "a["<<i<<"]="<<a[i] << std::endl;

    return 0;
}
```

---

## 13. Array basics - initialize arrays 5

```
#include <iostream>

int main()
{
    int m[2][3] = { {1,2,3}, {4,5,6} };

    for (int r = 0; r < 2; ++r)
    {
        for (int c = 0; c < 3; ++c)
            std::cout << m[r][c] << ' ';
        std::cout << "\n";
    }

    return 0;
}
```

---

## 14. Array basics - column sums

```cpp
#include <iostream>

int main()
{
    const int R=3, C=3;
    int m[R][C] = { {2,1,0}, {3,5,7}, {4,6,8} };

    for (int c = 0; c < C; ++c)
    {
        int sum = 0;
        for (int r = 0; r < R; ++r)
            sum += m[r][c];
        std::cout << "col " << c << " sum = " << sum << std::endl;
    }

    return 0;
}
```

---

## 15. Array basics - strings 1

```cpp
#include <iostream>

int main()
{
    char s[] = "HELLO";             // size = 6 including '\0'

    for (int i = 0; s[i] != '\0'; ++i)
        std::cout << s[i] << ' ';
    std::cout << "\n";

    s[1] = 'a';                     // overwrite a character
    std::cout << s << "\n";         // prints "HaLLO"

    return 0;
}
```

---

## 16. Array basics - strings 2

```cpp
#include <iostream>

int main()
{
    const char days[7][4] = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"};

    for (int i = 0; i < 7; ++i)
        std::cout << i << ": " << days[i] << "\n";

    std::cout << days[0][1] << "\n";
```

```
    // Access: days[0][1] is 'u', days[2] is "Tue"

    return 0;
}
```

---

## 17. Dice histogram

```
#include <iostream>

int main()
{
    int n;

    std::cout << "Enter number of rolls: ";
    std::cin >> n;
    if(n <= 0)
        return 0;

    int cnt[7];
    for(int i=0; i<7; ++i)
        cnt[i]=0;

    for(int i=0;i<n;++i)
    {
        int x;
        std::cin>>x;

        if(x>=1 && x<=6)
            ++cnt[x];
    }

    for(int face=1; face<=6; ++face)
    {
        std::cout << face << ": ";
        for(int k=0; k<cnt[face]; ++k)
            std::cout << '*';
        std::cout<<" (" << cnt[face] << ")\n";
    }

    return 0;
}
```

---

## 18. Sum of diagonal elements of a matrix

```
#include <iostream>

int main()
{
    int n;
```

```
    std::cout << "Enter a number = number of rows/columns (square): ";
    std::cin >> n;
    if(n <= 0)
        return 0;

    int a[25][25];
    if(n > 25)
        n = 25;
    for(int i = 0; i < n; ++i)
        for(int j = 0; j < n; ++j)
            std::cin >> a[i][j];

    int d1 = 0,d2 = 0;

    for(int i = 0; i < n; ++i)
    {
        d1 += a[i][i];
        d2 += a[i][n-1-i];
    }
    std::cout << "main diag=" << d1 << " other diag=" << d2 << std::endl;

    return 0;
}
```

---

## 19. Count vowels and consonants

```
#include <iostream>
#include <string>
#include <cctype>

int main()
{
    std::string s;

    std::cout<<"Line: ";
    std::getline(std::cin, s);

    int v = 0, c = 0;
    for (std::size_t i = 0; i < s.size(); ++i)
    {
        unsigned char ch = static_cast<unsigned char>(s[i]);
        if (std::isalpha(ch))
        {
            ch = static_cast<unsigned char>(std::tolower(ch));

            if (ch == 'a' || ch == 'e' || ch == 'i' || ch=='o' || ch == 'u')
                ++v;
            else
                ++c;
        }
    }
```

```
    std::cout << "vowels=" << v << " consonants=" << c << std::endl;

    return 0;
}
```

---

## 20. Manual string reverse

```
#include <iostream>
#include <string>

int main()
{
    std::string s;

    std::cout<<"Enter word: ";
    std::getline(std::cin, s);

    std::size_t j = s.size();
    if (j)
    {
        j--;
    }
    else
    {
        cout << "Empty string" << std::endl;
        return 0;
    }
    for (std::size_t i=0; i < j && j >= 0; ++i, --j)
    {
        char t=s[i];
        s[i]=s[j];
        s[j]=t;
    }
    std::cout << s << std::endl;

    return 0;
}
```

---