1. Recursion.

Task 1 (C++) We have to solve the following problem:

- We have an array of n integers tab[n]={tab[0],tab[1],…,tab[n-1]},
- Is there in an array tab number x (given as parameter)?

[p1.cpp]

```
#include <iostream>
#include <cstdlib>

using namespace std;
const int n=10;
int tab[n]={12,3,2,-7,44,5,1,0,-3};

void find(int tab[n], int left, int right, int x){

    if(left>right){
        cout<<"Element "<<x<<" was not found\n";
    }
    else{
        if(tab[left]==x){
            cout<<" Searched parameter has been found "<<x<<endl;
        }
        else
            find(tab,left+1,right,x);
    }
}

int main(){

find(tab,0,n-1,7);
find(tab,0,n-1,5);

system("pause");
return 0;

}
```

Task 2 (C++) Write a simple program that calculates the factorial of a given number in a recursive way.

[p2.cpp]

```
#include <iostream>
#include <cstdlib>

using namespace std;

unsigned long int factorial(int x){
if(x==0)
    return 1;
else
    return x* factorial(x-1);
}
```

```
// recurrence with auxiliary parameter
unsigned long int factorial2(int x, int temp=1){

if(x==0)
    return temp;
else
    return factorial2(x-1,x*temp);

}

int main(){

cout << " factorial(5)="<< factorial(5)<<endl;
cout << " factorial2(5)="<< factorial2(5)<<endl;

system("pause");
return 0;

}
```

Task 3 (C++) Write a program defining elements of the so-called the Fibonacci sequence. It occurs frequently in nature and is defined as follows:

fib(0)=0;
fib(1)=1,
fib(n)=fib(n-1) + fib(n-2), gdzie n>=2

The elements of this sequence are natural numbers and each element (with the exception of the first two) is the sum of previous two (ie. 1, 1, 2, 3, 5, 8, 13, ...).

[p3.cpp]

```
#include <iostream>
#include <cstdlib>
// The Fibonacci sequence
using namespace std;

unsigned long int fib(int x){

    if(x<2){
        return x;
    }
    else{
        return fib(x-1)+fib(x-2);
    }
}

int main(){

cout << "fib(6)="<<fib(6)<<endl;

system("pause");
return 0;

}
```

Task 3 a) Draw a tree of recursive calls for function fib(4) from a task **T3**.

Task 4 (C++) Write a program drawing a spiral (Fig. 1) recursively.

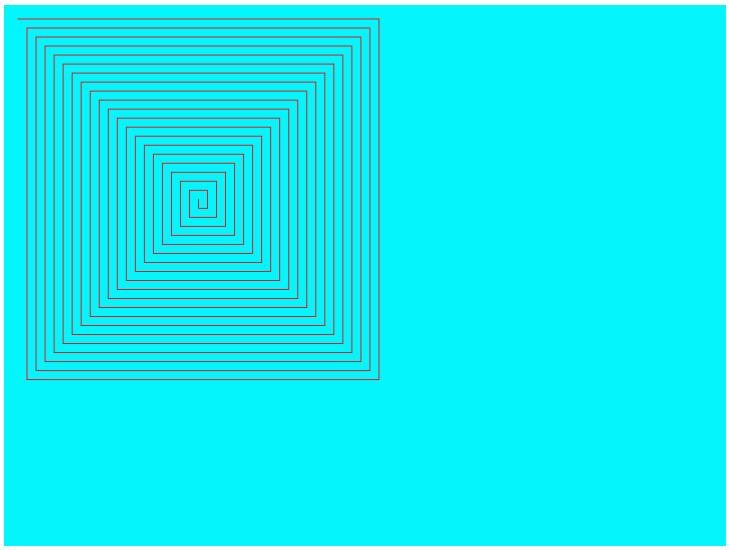

Fig 1.

Create a new project (allegro static):
New project -> MultiMedia -> Allegro application (static) [Project C++].

[main.cpp]

```
#include <allegro.h>
#define m 800
#define n 600

void init();
void deinit();
void spirala(int x, int y, int xl, int yl, int dx, int dy, int k);
char a[m+2][n+2], b[m+1][n+1];

BITMAP *bufor=NULL;

int main(){
      allegro_init();
      install_keyboard();
      set_color_depth(24);
      set_gfx_mode(GFX_AUTODETECT_WINDOWED,m,n,0,0);
      set_palette(default_palette);
      clear_to_color(screen,makecol24(0,244,255));
      bufor=create_bitmap(m,n);
    clear_to_color(bufor,makecol24(0,244,255));

    spiral(15,15,400,400,10,10,20);
    //blit(bufor,screen,0,0,0,0,m,n);

    while(!key[KEY_ESC]){
        readkey();
        if(key[KEY_F9]){save_bitmap("spirala.bmp", bufor,
default_palette);};
    }

    destroy_bitmap(bufor);
    deinit();
```

```
    return 0;
}
END_OF_MAIN()

void init() {
      int depth, res;
      allegro_init();
      depth = desktop_color_depth();
      if (depth == 0) depth = 32;
      set_color_depth(depth);
      res = set_gfx_mode(GFX_AUTODETECT_WINDOWED, 640, 480, 0, 0);
      if (res != 0) {
            allegro_message(allegro_error);
            exit(-1);
      }

      install_timer();
      install_keyboard();
      install_mouse();
      /* add other initializations here */
}

void deinit() {
      clear_keybuf();
      allegro_exit();
}

void spiral(int x, int y, int xl, int yl , int dx, int dy, int k){
    int c=0xff0000;
    if(k>0){
        hline(screen,x,y,x+xl,c);//x1,y1,x2
        rest(50);
        vline(screen,x+xl,y,y+yl,c);//x1,y1,y2
        rest(50);
        hline(screen,x+dx,y+yl,x+xl,c);
        rest(50);
        vline(screen,x+dx,y+dy,y+yl,c);
        rest(50);
        x=x+dx;y=y+dy;xl=xl-2*dx;yl=yl-2*dy;
        spirala(x,y,xl,yl,dx,dy,k-1);
    }
}
```

Task 5 (C++) Write a program for reversal an array (integer elements) in a recursive way. [p5.cpp]

```
#include <iostream>
#include <cstdlib>

using namespace std;
int t[10]={0,1,2,3,4,5,6,7,8,9};

void reverse(int * m, int left, int right){
    if(right>left){
        int temp=m[left];
        m[left]=m[right];
        m[right]=temp;
        reverse(m,left+1,right-1);
    }
```

```
}

int main(){

cout << "Array"<<" t[] before reversal:\n";
for(int i=0;i<10;i+=1){
cout<<"t["<<i<<"]="<<t[i]<<",";
}

reverse(t,0,9);
cout << "\nArray"<<" t[] after reversal:\n";
for(int i=0;i<10;i+=1){
cout<<"t["<<i<<"]="<<t[i]<<",";
}
cout<<endl;

system("pause");
return 0;

}
```

Task 6 (C++) Draw a tree of recursive calls of McCarthy function:

```
unsigned long int McCarthy(int x){
    if(x>100){
        return x-10;
    }
    else{
        return McCarthy(McCarthy(x+11));
    }
}
```

for the number x = 100 and x = 99. Then check out the results by writing a program.

[p6.cpp]

```
#include <iostream>
#include <cstdlib>

using namespace std;
int temp=0;

unsigned long int McCarthy(int x){
    temp++;
    if(x>100){
        return x-10;
    }
    else{
        return McCarthy(McCarthy(x+11));
    }
}

int main(){

for(int i=90;i<=100;i+=1){
temp=0;
cout << "McCarthy("<<i<<")="<<McCarthy(i)<<" ";
cout << temp << endl;
```

```
}
system("pause");
return 0;

}
```

Task 7 (C++) Write a program that for a given positive integer number print its representation in binary system.

Tip: use an known dividing algorithm by the base system.
For example, for a number 13:
13 : 2 = 6 + 1,
6 : 2 = 3 + 0,
3 : 2 = 1 + 1,
1 : 2 = **0** + 1. (**0** – end of the program).

[p7.cpp]

```
#include <iostream>
#include <cstdlib>

//binary representation

using namespace std;

void binary(int a){
    if(a>0){
        if((a%2)==0){
            a/=2;
            binary(a);
            cout<<0;
        }
        else{
            a/=2;
            binary(a);
            cout<<1;
        }
    }
}

void binary2(int a){
    if(a!=0){
        binary(a/2);    //a modulo 2
        cout<<a%2;      // remainder of the division by 2
    }
}

int main(){

int temp;
cout<<"Conversion to binary \nEnter a positive integer:\n";
cin>>temp;

binary(temp);
cout<<endl;

binary2(temp);
```

```
cout<<endl;

system("pause");
return 0;

}
```

Task 8 (C++) Write a program drawing squares recursively (Fig. 2).

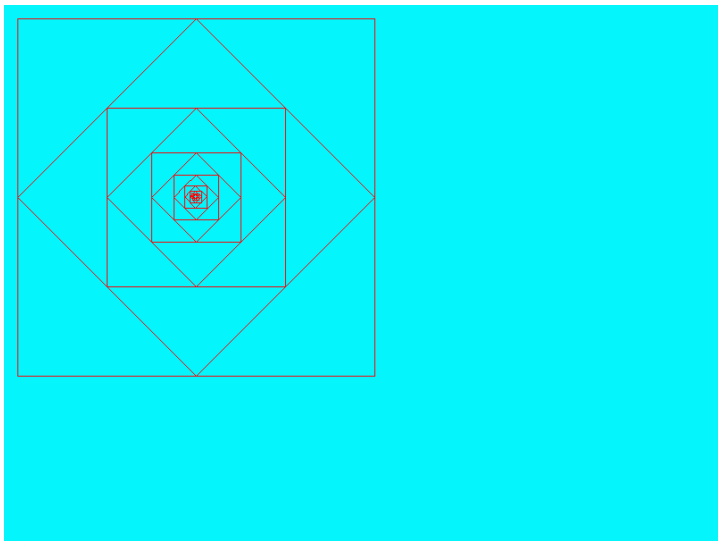

Fig 2.

Tip: modify the function responsible for drawing spiral of job **T4**.
Important is to choose the right place of recursive call..

[p8.cpp]

```
#include <allegro.h>
#define m 800
#define n 600

void init();
void deinit();
void kwadraty(int x, int y, int xl, int yl);
char a[m+2][n+2], b[m+1][n+1];

BITMAP *bufor=NULL;

int main(){
      allegro_init();
      install_keyboard();
      set_color_depth(24);
      set_gfx_mode(GFX_AUTODETECT_WINDOWED,m,n,0,0);
      set_palette(default_palette);
      clear_to_color(screen,makecol24(0,244,255));
      bufor=create_bitmap(m,n);
    clear_to_color(bufor,makecol24(0,244,255));

    squares(15,15,400,400);
    blit(bufor,screen,0,0,0,0,m,n);

    while(!key[KEY_ESC]){
```

```
        readkey();
        if(key[KEY_F9]){save_bitmap("kwadraty.bmp", bufor,
      default_palette);};
    }

    destroy_bitmap(bufor);
    deinit();
    return 0;
}
END_OF_MAIN()

void init() {
      int depth, res;
      allegro_init();
      depth = desktop_color_depth();
      if (depth == 0) depth = 32;
      set_color_depth(depth);
      res = set_gfx_mode(GFX_AUTODETECT_WINDOWED, 640, 480, 0, 0);
      if (res != 0) {
            allegro_message(allegro_error);
            exit(-1);
      }

      install_timer();
      install_keyboard();
      install_mouse();
      /* add other initializations here */
}

void deinit() {
      clear_keybuf();
      allegro_exit();
}

void squares(int x, int y, int xl, int yl){
    int c=0xff0000;
    if(xl>2&&yl>2){
        hline(bufor,x,y,x+xl,c);//x1,y1,x2
        vline(bufor,x+xl,y,y+yl,c);//x1,y1,y2
        hline(bufor,x,y+yl,x+xl,c);
        vline(bufor,x,y+yl,y+yl/2,c);

        line(bufor,x,y+yl/2,x+xl/2,y+yl,c);
        line(bufor,x+xl/2,y+yl,x+xl,y+yl/2,c);
        line(bufor,x+xl,y+yl/2,x+xl/2,y,c);
        line(bufor,x+xl/2,y,x+xl/4,y+yl/4,c);

        x=x+xl/4;y=y+yl/4;xl=xl-xl/2;yl=yl-yl/2;
        kwadraty(x,y,xl,yl);

        xl=2*xl;yl=2*yl;x=x-xl/4;y=y-yl/4;
        line(bufor,x+xl/4,y+yl/4,x,y+yl/2,c);
        line(bufor,x,y+yl/2,x,y,c);
    }
}
```