

main.cxx

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <string>
5 #include <utils.hxx>
6 #include <ui.hxx>
7
8
9 int main(int argc, char** argv) {
10     // flags for program arguments
11     bool upAlphaFlag = false;
12     bool lowAlphaFlag = false;
13     bool numFlag = false;
14     bool specialCharFlag = false;
15     int length = -1;
16     int arg;
17     int state = 0;
18
19     while ((arg = getopt (argc, argv, "Aanshl:")) != -1) {
20         switch (arg) {
21             case 'A':
22                 upAlphaFlag = true;
23                 break;
24             case 'a':
25                 lowAlphaFlag = true;
26                 break;
27             case 'n':
28                 numFlag = true;
29                 break;
30             case 's':
31                 specialCharFlag = true;
32                 break;
33             case 'l':
34                 length = std::atoi(optarg);
35                 break;
36             case 'h':
37                 printhelp();
38                 return 0;
39             case '?':
40                 if (optopt == 'l') {
41                     printf("Error: No length specified\n");
42                 } else {
43                     printf("Error: Unknown value: -%c\n", optopt);
44                 }
45                 return 1;
46             default:
47                 printhelp();
48                 abort();
49         }
50     }
51
52     if (length < 0 && checkFlags(upAlphaFlag, lowAlphaFlag, numFlag,
53     , specialCharFlag) == false) {
54         state = rungui(argc, argv);
55     } else {
```

```
55     state = runcui(length, upAlphaFlag, lowAlphaFlag, numFlag,  
56     specialCharFlag);  
57     }  
58     return state;  
59 }
```

passgen.hxx

```
1 #ifndef PASSGEN
2 #define PASSGEN
3
4 #include <stdlib.h>
5 #include <stdio.h>
6
7 namespace PassGen {
8     char* getUpperAlpha();
9     char* getLowerAlpha();
10    char* getNumber();
11    char* getSpecialChars();
12    char* passGen(char* charList, int len);
13 }
14
15 #endif // PASSGEN
```

passgen.cxx

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #include <utils.hxx>
6 #include <passgen.hxx>
7
8 // getSpecialChars - get the lower case alphabets
9 // void : takes nothing
10 // return (char*) : the string with all lower case alphabets in
    standard ASCII
11 char* PassGen::getLowerAlpha() {
12     char* output = new char[26]; // 26 letters
13     if (output == NULL) {return 0;} // check if memory allocation
        is failed
14     int offset = 97; // 97th letter in ASCII (a)
15     for (int i = 0; i < 26; i++) {
16         output[i] = offset + i;
17     }
18     return output;
19 }
20
21 // getUpperAlpha - get the upper case alphabets
22 // void : takes nothing
23 // return (char*) : the string with all upper case alphabets in
    standard ASCII
24 char* PassGen::getUpperAlpha() {
25     char* output = new char[26]; // 26 letters
26     if (output == NULL) {return 0;} // check if memory allocation
        is failed
27     int offset = 65; // 65th letter in ASCII (A)
28     for (int i = 0; i < 26; i++) {
29         output[i] = offset + i;
30     }
31     return output;
32 }
33
34 // getSpecialChars - get the numbers
35 // void : takes nothing
36 // return (char*) : the string with all numbers in standard ASCII
37 char* PassGen::getNumber() {
38     char* output = new char[10]; // 10 letters
39     if (output == NULL) {return 0;} // check if memory allocation
        is failed
40     int offset = 48; // 48th letter in ASCII (0)
41     for (int i = 0; i < 10; i++) {
42         output[i] = offset + i;
43     }
44     return output;
45 }
46
47
48 // getSpecialChars - get the special characters
49 // void : takes nothing
50 // return (char*) : the string with all special characters in
```

```

51     standard ASCII
52 char* PassGen::getSpecialChars() {
53     char* output = new char[42]; // 42 symbols
54     if (output == NULL) {return 0;} // check if memory allocation
55     is failed
56     int offset = 33; // 33rd letter in ASCII (!)
57     int listOffset = 0;
58     int i;
59     // ASCII range of 33 - 64 (32 symbols)
60     for (i = 0; i < 32; i++) {
61         output[i] = offset + i;
62     }
63     listOffset = 32;
64     offset = 91;
65     // ASCII range of 91 - 96 (6 symbols)
66     for (i = 0; i < 6; i++) {
67         output[i + listOffset] = offset + i;
68     }
69     listOffset = 38;
70     offset = 123;
71     // ASCII range of 123 - 126 (4 symbols)
72     for (i = 0; i < 4; i++) {
73         output[i + listOffset] = offset + i;
74     }
75     return output;
76 }
77
78 // passGen - Password Generator
79 // charList (char*) : list of char to be used in password
80 // generation
81 // len (int) : length of password
82 // return (char*) : the generated password
83 char* PassGen::passGen(char *charList, const int len) {
84     std::srand(time(nullptr));
85     unsigned int index;
86     char* output = new char[len+1]; // length of password + 1
87     terminating char
88     if (output == NULL) {return 0;} // return 0 on the failiure of
89     memory allocation
90     for (int i = 0; i <= len; i++) {
91         if (i == len) {output[i] = charList[strSize(charList)];}
92         else {
93             index = std::rand()%(strSize(charList));
94             output[i] = charList[index];
95         }
96     }
97     return output;
98 }

```

utils.hxx

```
1 #ifndef UTILS
2 #define UTILS
3
4 #include <stdlib.h>
5
6 inline int strSize(char* a) {
7     int out = 0;
8     int i = 0;
9     while (a[i] != 0) {
10         i++;
11         out++;
12     }
13     return out;
14 }
15
16 inline void printhelp() {
17     printf("APCSPCreateTask - Random Password Generator\n\n");
18     printf("[Usage]: APCSPCreateTask [-A -a -n -s] -l <length>\n\n");
19     printf("[Arguments]:\n\n");
20     printf("\t-A : include upper case alphabets in password\n\n");
21     printf("\t-a : include lower case alphabets in password\n\n");
22     printf("\t-n : include numbers in password\n\n");
23     printf("\t-s : include special characters in password\n\n");
24     printf("\t-l <number> : set the length of the password\n\n");
25     printf("\t-h : print this help\n\n");
26 }
27
28 inline bool checkFlags(bool up, bool low, bool num, bool spec) {
29     int count = 0;
30     if (up == true) {count++;}
31     if (low == true) {count++;}
32     if (num == true) {count++;}
33     if (spec == true) {count++;}
34     return count > 0 ? true : false;
35 }
36
37 #endif // UTILS
```

ui.hxx

```
1 #ifndef UI
2 #define UI
3
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #include <utils.hxx>
8 #include <passgen.hxx>
9 #include <gtkui.hxx>
10 using namespace PassGen;
11
12 int runcui(int len, bool up, bool low, bool num, bool special) {
13
14     std::string input;
15
16     if (len < 0) {
17         printhelp();
18         printf("\nError: Length not specified\n");
19         return 1;
20     }
21
22     if (checkFlags(up, low, num, special) == false) {
23         printhelp();
24         printf("\nError: No character flag(s) specified\n");
25         return 1;
26     }
27
28     if (up == true) {input += getUpperAlpha();}
29     if (low == true) {input += getLowerAlpha();}
30     if (num == true) {input += getNumber();}
31     if (special == true) {input += getSpecialChars();}
32
33     char *cInput = new char[input.length() + 1];
34     strcpy(cInput, input.c_str());
35     char *out = passGen(cInput, len);
36     std::cout << out << std::endl;
37     return 0;
38 }
39
40 int rungui(int argc, char** argv) {
41     auto app = Gtk::Application::create("io.github.kenryus");
42     return app->make_window_and_run<PassGenUI>(argc, argv);
43 }
44
45 #endif // UI
```

gtkui.hxx

```
1 #ifndef GTKUI
2 #define GTKUI
3
4 #include <gtkmm.h>
5
6 class PassGenUI : public Gtk::Window
7 {
8     public:
9         PassGenUI();
10        ~PassGenUI();
11        void on_button_clicked();
12     private:
13         Gtk::Button m_button;
14 };
15
16 #endif // GTKUI
```


gtkui.cxx

```
1 #include <gtkui.hxx>
2 #include <iostream>
3
4 PassGenUI::PassGenUI()
5 : m_button("Hello") {
6     m_button.set_margin(10);
7     m_button.signal_clicked().connect(sigc::mem_fun(*this, &
8     PassGenUI::on_button_clicked));
9     set_child(m_button);
10 }
11
12 PassGenUI::~PassGenUI() {
13 }
14
15 void PassGenUI::on_button_clicked() {
16     std::cout << "Hello" << std::endl;
17 }
```