1)ABDE are all correct because they both start with “a” and ends with “a” and [ab] means that you can have any combination of a or b in between. * allow for 0 or more occurrences. So [ab]*0 or [ab]*n meaning you can have any combination/amount of the letter “a” or “b” in between the 2 “a”

2)The correct answer is AD because (BC)? Is optiontal, because ? matches zero or one occurrence of the single preceding character, and in this case, its the group (BC) so we can have 1 occurrence of bc or 0. The only two that is allowed is if bc is not there so only a or abc thus, the rest are wrong due to the addition strings

3)a “.” symbol means any character for our first character and we are allowed 0 or any more occurrences of [ind] characters because of * which is greedy because there is an unlimited amount of any letters in the list group. A and C are all correct answers because they all contain some form of nd, in,or ind and no extra letters afterwards.

4)C and E are correct because each string contains a lowercase letter because the strings must be together and can have 1 or 2 lowercase letters. The rest are wrong because they have a string + and if we want a + as a string, we need to use \x

5)[a-z] ask for lowercase number, \+ ask for “+” string, followed by another lowercase number and + outside allows for unlimited occurrence of the previous group, which would be the lowercase letter with +. Therefore, it's asking for lowercase+lowercase or lowercase+lowercase+lowercase... The only correct answer is A and E

6)ABCD is correct because all of the above are correct since they include [a-z] lowercase characters, no spaces due to +, and contain a string of “.”,”!”,”or “?” at the end of the string. The only one that is wrong, is E assuming that it is a statement/string because it contains space. However, if E is not a statement and includes ABCD, then it would have been correct.

7)CDE is correct because the strings can have 0 or 1 occurrences of “hot ” due to the “?” but it must have at least 1 “very” near the beginning of string and weather at the end and can have either bad,stormy, or good due to the usage of the or function |. A is missing very and B is missing either bad,good, or stormy.

8)E is correct because “-?” means you can have either a “-” character string or not. [0-9]* means any occurrences of a number from 0 to 9. The “.” as a string is optional because of \.? Where the \ makes the “.” a string and ? allows for 0 or 1 occurrences. All the above 4 choices are correct as are most of the numbers and “.” are optional so not necessary, therefore, a single “.” also works.

Part 2

2) Answer: ^[a-zA-Z _][a-zA-Z _]{0,9}
Explanation: ^[a-zA-Z_] ensures that the first character starts with any letter or underscore where ^ is our pointer for the first char. Afterwards [a-zA-Z_] indicates any letters but {0,9} specifies the next 9 characters because we already have a first character to make 10 characters total

3) Answer:(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])[A-Za-z\d]{4,10}$
and ?=* sets the condition to include at least one of which in this case, we [A-Z] and [a-z] and the 0-9 indicates digits. Then, [A-Za-z\d]{4,10} specifies that we can have any upper or lower case letters or numbers between 4 through 10 as our requirement.

4) Answer: (0?[1-9]|1[0-2])\/(0?[1-9]|[12]\d|30|31)\/(\d{4})
(0?[1-9]|1[0-2]) matches the month with 0? As optional for when the month is single digits such as february and 1-9 to account for january to september. | is a or statement, which states that or we can have a value from 0,1,2 to in the ones section using 1 to specify that to account for October, November, and december. \/ specifies that we get a “/” as a string. The date shares a similar logic to our month except we added extra or condition to account for the fact that some months do not have 30 or 31. Year I just used \d{4} to allow any number of digits for the next 4 characters, which would complete the year assignment.

5) answer: \d{3}-\d{3}-\d{4}| \(\d{3}\)-\d{3}-\d{4}
\d{3} creates the next 3 digits, - separates,and then the \d{4} calls for the next 4 digits. Afterwords, I use the | which is the or operand to take in (\d{3}) to create (###) and the or will take either one depending on which one was provided.

6)

Main.c 438pqcgmk AI NEW C RUN

```c
#include <stdio.h>
int main()
{
    printf("*\n* *\n* * *\n* * * *");
}
```

STDIN

Input for the program (Optional)

Output:

```
*
* *
* * *
* * * *
```

7)

Circle Area 438vc28q7 AI NEW C RUN

```c
#include <stdio.h>
#include <math.h>
float distance(int x1, int y1, int x2, int y2) {
    return sqrtf(powf(x2 - x1, 2) + powf(y2 - y1, 2));
}

int main() {
    int x1 = 2, y1 = 2, x2 = 5, y2 = 6;
    float diameter = distance(x1, y1, x2, y2);
    float radius = diameter / 2;
    float perimeter = 2 * M_PI * radius;
    float area = M_PI * powf(radius, 2);
    printf("Perimeter %.3f\n", perimeter);
    printf("Circle Area: %.3f\n", area);
    return 0;
}
```

STDIN

Input for the program (Optional)

Output:

```
Perimeter 15.708
Circle Area: 19.635
```