Grupo I:

Existiam 7 erros:

- Declaração termina com ;
- Condicional 1º ciclo for: i<n (ou i<n-1)
- Incremento 1º ciclo for: i++
- Inicialização 2º ciclo: j=i+1;
- Incremento 2º ciclo for: j++
- Condicional if: v[i]>v[j] (ou v[i]>=v[j])
- 2ª atribuição: v[i]=v[j];

Critérios:

- 0,5 por cada erro detectado, até ao máximo de 3 valores (basta identificar 6 dos 7 erros).
- 0,5 por erro introduzido não existente (cancela um erro bem identificado)
- Houve quem tivesse colocado o código corrigido, e quem indique os erros. Ambas as formas são válidas.
- 0,3 por um erro identificado mas não corrigido
- 0,2 erro detectado, mas corrigido de forma incorreta
- A substituição do código por outro, com a funcionalidade implementada, não responde ao solicitado, que é a identificação dos erros no código fornecido. Nestes casos a pergunta poderá ser valorizada até metade, e no caso de serem apontados erros ao código original, é valorizada essa parte da resposta.

Grupo II:

Critérios positivos (soma de 0 a 1,5 valores):

- + 0,5 - assinatura correta
- + 0,5 - retorno da função correto
- + 0,5 - lógica relativamente correta

Negativos (subtrai de 3 a 1,5)

- - 0,5 - erro no código (expressão numérica/lógica incorreta; instrução incorreta)
- - 0 - um erro sintático (inclui troca de aspas por plicas e vice-versa)
- - 0,5 - dois ou mais erros sintáticos menores.

Situações normalizadas:

- -0,5 não terminar a string com um caracter 0
- -0,25 troca de K por W
- -0.5 colocar rand(W), quando rand não tem argumentos, em vez de rand()%W
- -0,25 retorno da função distinto de void, não retornando nada (c.c. é a situação seguinte)
- -0,5 return e tipo de retorno (um só erro), quando a função não deve retornar nada
- -0,25 utilizar srand()%W em vez de rand()%W
- -0,5 não somar 'A'
- 0 - inicialização de uma string com o alfabeto, colocando apenas as letras iniciais
- -0,25 chamar rand%W em vez de rand()%W
- -0,25 utilizar rand()%(W-1) em vez de rand()%W

Solução considerada mais simples:

```
void GerarChave(char *chave, int K, int W) {
	int i;
	for (i = 0; i<K; i++)
		chave[i] = 'A' + rand() % W;
	chave[K] = 0;
}
```

Grupo III

Critérios positivos/negativos idêntico ao grupo anterior.

Situações normalizadas:

- 1,5 Calculo completamente operacional, com brancas a zero, e pretas com o valor correto (nenhum outro critério é contabilizado, neste caso);
- 2 Funcionalidade das pretas corretas, e tentativa das brancas
- -0,5 Parcialmente correto (estaria certo se existir apenas uma só letra)
- +0,25 assinatura parcialmente completa (falha em um ou dois dos parâmetros)
- +0,5 calculo da variável pretas correto (contagem de caracteres iguais, mesmo com falhas de outro tipo)
- 0 penalização por alterar a chave/aposta da entrada inicial (eliminar as letras utilizadas ao contar pretas/brancas)

Solução considerada mais simples e completamente correta, não alterando as variáveis de entrada (o enunciado revelava a solução em processar primeiro as pretas, depois as brancas):

```
void AnaliseAposta(char *chave, char *aposta, int K, int W, int *pretas, int *brancas) {
	int i;
	char dupChave[MAXSTR], dupAposta[MAXSTR], *pt;
	*pretas = 0;
	*brancas = 0;
	// atualizar as pretas
	for (i = 0; i<K; i++) {
		if (chave[i] == aposta[i]) {
			(*pretas)++;
			dupChave[i] = dupAposta[i] = ' '; // já foi utilizada, não copiar
		}
		else {
			dupChave[i] = chave[i]; // pode existir no local errado
			dupAposta[i] = aposta[i];
		}
	}
	// calcular as brancas
	dupAposta[K] = 0;
	dupChave[K] = 0;
	for (i = 0; i<K; i++)
		if (dupChave[i] != ' ') {
			pt = strchr(dupAposta, dupChave[i]);
			if (pt != NULL) {
				// certo no local errado
				(*brancas)++;
				*pt = ' '; // assegurar que não é contabilizada novamente
			}
		}
}
```

Grupo IV

Critérios positivos/negativos idêntico ao grupo anterior.

Situações normalizadas:

- -0,5 solicitação de K e W dentro do ciclo (deve ser fora)

Houve quem tivesse reproduzido o grupo 2 e 3, não era necessário.

Em algumas provas, este grupo ficou por fazer, e era mais fácil que o grupo 2 e 3.

Solução considerada mais simples:

```
int main() {
	int K, W, brancas, pretas;
	char chave[MAXSTR], aposta[MAXSTR];
	srand(1);
	printf(""K: "");
	scanf(""%d"", &K);
	printf(""W: "");
	scanf(""%d"", &W);
	gets(aposta);
	GerarChave(chave, K, W);
	do {
		printf(""\nAposta: "");
		gets(aposta);
		AnaliseAposta(chave, aposta, K, W, &pretas, &brancas);
		printf("" p%d b%d"", pretas, brancas);
	} while (pretas!=K);
}
```