```
//Q1.1
//the function take a string representing a number and converts it to the corresponding integer
//fix:
int func(char* str){
 int sigen = 1;
if(*str == '-'){
	sigen = -1;
	str++;
}
 int res = *str;
 //str++; // remove this
 while(*str){
 res*=10;
 //res+= *str;
 res+= *str - '0'; //the '0' is a char
 str++;
 }
 return res*sigen;
}

//Q1.2
123456789
9

//Q1.3

int length(int num) {
	int count = 0;
	while(num>0) {
	count++;
	num /=10;
	}
	return count;
}

char* toString(int num) {
	int len = length(num) , tmp = num,pow=0,i=0;
	char* str = (char*)malloc((len+1)*sizeof(char));
	if (str == NULL) {
		printf("allocation error");
		exit(1);
	}
	while(tmp) {
		pow = pow(10,len-i);
		tmp = tmp/pow;
		str[i] = tmp - '0';
		i++;
		tmp=num%pow;
	}
	str[i] = 0; //same as '\0'
	return str; // we must not free str.
}

//Q2

typedef struct _EMPLOYEE {
	int id;
	char* name;
	struct _EMPLOYEE* subordinates[5];
}employee,*pemployee;

pemployee new_emp(char* name, int id){
	pemployee n = (pemployee)malloc(sizeof(employee));
	if (n == NULL) {
		printf("allocation error");
```

```
		exit(1);
	}
	n->id = id;
	n->name = (char*)malloc((strlen(name) +1)*sizeof(char)); //we must allocate new space for the string;
	if (n->name == NULL) {
		printf("allocation error");
		free(n);
		exit(1);
	}
	for (int i=0 ; i<5;i++) n->subordinates[i] = NULL;
	return n;
}

pemployee find_manager(pemployee manager , id) {
	if (manager == NULL) return NULL;
	if (manager->id == id) return manager;
	for (int i=0; i<5;i++){
		pemployee found = find_manager(manager->subordinates[i],id);
		if(found) return found;
	}
	return NULL;
	}

void add(pemployee* company , pemployee emp , int manager_id) {
	if (manager_id == -1) {
		*company = emp; //this is the CEO. he has no manager. we didn't deduct points if you didn't to this.
		printf("employee added successfully");
		return;
	}
	pemployee manager = find_manager(*company , id);
	int i=0;
	for ( ; i<5 ;i++ ) {
		if(manager->subordinates[i] == NULL) {
			manager->subordinates[i] = emp;
			break;
		}
	}
	if (i == 5) printf("this manager can't have any more employees");
	else printf("employee added successfully");
	return;
}

//Q3
# define NUM_LETTERS 'z'-'a' +1
int max_new_increase_substr(char* string) {
	int max_letter_num=0;
	while (*string) {
		int letter_num= max_increase_with_memory_substr_from_start(string);
		if (max_letter_num < letter_num) {
			max_letter_num=letter_num; max_letter_num=letter_num;
			}
		string++;
		}
	return max_letter_num;
}

int max_increase_with_memory_substr_from_start(char* string) {
	int used_letters[NUM_LETTERS] = {};
	char max_letter = *string;
	int max_letter_num = 0;
	while (*string) {
		if (*string > max_letter) {
			max_letter = *string;
			used_letters[*string -'a'] = 1;
			max_letter_num++;
```

```
                }
            else if(used_letters[*string -'a']) {
                max_letter_num++;
            }
            else {
                return max_letter_num;
            }
            string++;
        }
        return max_letter_num;
}


int main() {
        char c,*str = null;
        int srt_len = 0;
        while(EOF != scanf("%c",&c)) {
            if (str == null) {
                str = (char*)malloc(sizeof(char));
                if (str == null) {
                    printf("allocation error");
                    exit(1);
                }
                srt_len++;
            }
            else {
                str = realloc(str,++srt_len*sizeof(char))
                if (str == null) {
                    printf("allocation error");
                    exit(1);
                }
            }
            str[srt_len-1] = c;
        }
        printf("max new increase substring= %d",max_new_increase_substr(str));
        free(str);
        return 0;
}
```